2D Finite-Difference Wavefield Modelling

Jan Thorbecke

February 16, 2017

Contents

0   Getting Started

0.1   Installation

The software, downloaded as a gzipped tar archive, can be extracted in a directory of your choice, e.g., by typing

```
> tar -xvfz OpenSource.tgz
```

at the terminal command line of a Unix based operating system.

The code is designed to run on current Unix-based or Unix-like system, such as Linux, Sun's Solaris, Apple's OS-X or IBM's AIX. However, the code has not been tested on any version of Windows. It is certainly possible to make it run on a Windows platform using the appropriate tools, but do not expect it to work simply out-of-the-box. For running the code in a Windows environment one could make use of Cygwin. Cygwin is a virtual Linux environment running in a Windows operating system.

The package extracts itself into a directory `OpenSource`, with the following sub-directories:

- bin

- lib

- doc

- include

- fdelmodc

- utils

- FFTlib

The `README` file in that directory contains some of the quick-start information given here in condensed format. The FFTlib directory does not contain a main program and contains source code to build a library (`libgenfft.a`) with Fourier transformation functions. The fdelmodc and utils directories include all files needed to compile and link the executables build in that directory. This means that some of the source files in the fdelmodc and utils directory are the same. This has been done to make the compilation procedure less complicated. Section A of this manual contains a brief (one-sentence) explanation of the meaning of all the files in the source tree of this package. The source code is in continuous development to add new features and solve bugs. The latest version of the source code and manual can always be found at:

http://www.xs4all.nl/ janth/Software/Software.html.

The code is used by many different people and when somebody requests a new option for the code (for example place receivers in a circle) then I will try to implement and test the new functionality and put the updated source (and manual) on the website as soon as it is ready and tested.

0.2   Compilation and Linking

1. To compile and link the code you first have to set the ROOT variable in the Make_include file which can be found in the directory where you have found this README.

2. Check the compiler and CFLAGS options in the file Make_include and adapt to the system you are using. The default options are set for a the GNU C compiler on a Linux system. A Fortran or g++ compiler is not needed to compile the code. The compilation of the source code has been tested with several versions of GNU and Intel compilers.

3. If the compiler options are set in the Make_include file you can type

```
> make
```

and the Makefile will compile and link the source code in the directories:

- FFT library

- fdelmodc

- utils

The compiled FFT library will be placed in the `lib/` directory, the executables in the `bin/` directory and the include file of the FFT library in the `include/` directory.
To use the executables don't forget to include the pathname in your PATH:

```
bash:
export PATH='path_to_this_directory'/bin:$PATH:
csh:
setenv PATH 'path_to_this_directory'/bin:$PATH:
```

On Linux systems using the bash shell you can put the `export PATH='path_to_this_directory'/bin:$PATH:` setting in `$HOME/.bashrc`, to set it every time you login.
Other useful make commands are:

- `make clean`: removes all object files, but leaves libraries and executables

- `make realclean`: removes also object files, libraries and executables.

0.3   Running examples

Important note: The examples and demo scripts make use programs of Seismic Unix (SU). Please make sure that SU is compiled without XDR: the XDR flag (`-DSUXDR`) in $CWPROOT/Makefile.config must **NOT** be set in compiling SU. The SU output files of fdelmodc are all base on local IEEE data. When the XDR flag is set in SU you have to convert the output files of fdelmodc (and all the programs in the utils directory: basop, fconv, extendmodel, makemod, makewave) with `suoldtonew`, before using SU programs.
If the compilation has finished without errors and produced an executable called fdelmodc you can run the demo programs by running. For example the script

```
> ./fdelmodc_plane.scr
```

in the directory `fdelmodc/demo/`. The results of this script are discussed in section 7.1. The `fdelmodc/demo/` directory contains scripts to demonstrate the different possibilities of the modeling program. Most of the scripts in the demo directory can re-produce the figures used in this manual. The examples section 7 contains also detailed explanations of the other demo scripts.
To reproduce the Figures shown in the GEOPHYICS manuscript "Finite-difference modeling experiments for seismic interferometry" (Thorbecke and Draganov, 2011) the scripts in `fdelmodc/FiguresPaper/` directory can be used. Please read the README in the FiguresPaper directory for more instructions and guidelines.
To clean-up all the produced output files in the `fdelmodc/demo/` and `fdelmodc/FiguresPaper/` directory you can run the `clean` script in those directories.

1   Introduction to Finite-Difference

The program `fdelmodc` can be used to model waves conforming the 2D wave equation in different media. This manual does not give a detailed overview about finite-difference modelling and only briefly explains the four different Finite-Difference (FD) schemes implemented in the program `fdelmodc`. More important are the (im)-possibilities of the program, and a detailed explanation is given how to use the parameters together with certain specific implementation issues a user must be aware of. There are already many programs available to model the 2D wave equation, and one might ask why write another one? The program `fdelmodc` is open source, makes use of the Seismic Unix (SU) parameter interface and output files, and specially aims at the modelling of measurements used for Seismic Interferometry. This means that noisy source signals at random source positions can be modeled for very long recording times using only one program.

The first four sections after the introduction describe the four implemented schemes; acoustic, visco acoustic, elastic, and visco elastic. In section 6 the program parameters are described and in section 7 examples are given how the program can be applied and demonstrates the possibilities of the program. The remainder of this introduction explains the finite-difference approximations for the derivatives used in the first-order systems governing the wave equation, and how the discretization must be chosen for stable and dispersion-free modelling results.

The program `fdelmodc` computes a solution of the 2D wave equation by approximating the derivatives in the wave equation by finite-differences. The wave equation is defined through the first-order linearized systems of Newton's and Hooke's law. For an acoustic medium the equations are given by;

$$
\begin{aligned}
\frac{\partial V_x}{\partial t} &= -\frac{1}{\rho}\frac{\partial P}{\partial x}, \\
\frac{\partial V_z}{\partial t} &= -\frac{1}{\rho}\frac{\partial P}{\partial z}, \\
\frac{\partial P}{\partial t} &= -\frac{1}{\kappa}\{\frac{\partial V_x}{\partial x}+\frac{\partial V_z}{\partial z}\},
\end{aligned}
\tag{1}
$$

where V_x, V_z are the particle velocity components in the x and z-direction, respectively, P the acoustic pressure, ρ is the density of the medium and κ the compressibility.

The first-order derivatives in the spatial coordinates (lateral position x and depth position z) are approximated by a so-called centralized 4'th order Crank-Nicolson approximation,

$$
\frac{\partial P}{\partial_x} \approx \frac{-P((i+\frac{3}{2})\Delta x)+27P((i+\frac{1}{2})\Delta x)-27P((i-\frac{1}{2})\Delta x)+P((i-\frac{3}{2})\Delta x)}{24\Delta x}
\tag{2}
$$

the first order derivative in time is approximated by a 2th order scheme:

$$
\frac{\partial P}{\partial_t} \approx \frac{P((i+\frac{1}{2})\Delta t)-P((i-\frac{1}{2})\Delta t)}{\Delta t}.
\tag{3}
$$

These approximations can be derived from linear combination of different Taylor expansions (Fornberg, 1988):

$$
P(x+\Delta x) \approx P(x)+\frac{\Delta x}{1!}\frac{\partial P}{\partial x}+\frac{\Delta x^2}{2!}\frac{\partial^2 P}{\partial x^2}+\frac{\Delta x^3}{3!}\frac{\partial^3 P}{\partial x^3}+\mathcal{O}\Delta x^4
\tag{4}
$$

For example, a 4th order approximation of a first-order derivative, used in the implemented staggered grid, can be derived from four Taylor expansions on 4 points centered around $x = 0$:

$$
\begin{aligned}
P(x+\frac{\Delta x}{2}) &\approx P(x)+\frac{\Delta x}{2}\frac{\partial P}{\partial x}+\frac{\Delta x^2}{8}\frac{\partial^2 P}{\partial x^2}+\frac{\Delta x^3}{24}\frac{\partial^3 P}{\partial x^3}+\frac{\Delta x^4}{96}\frac{\partial^4 P}{\partial x^4}+\mathcal{O}\Delta x^5 \\
P(x-\frac{\Delta x}{2}) &\approx P(x)-\frac{\Delta x}{2}\frac{\partial P}{\partial x}+\frac{\Delta x^2}{8}\frac{\partial^2 P}{\partial x^2}-\frac{\Delta x^3}{24}\frac{\partial^3 P}{\partial x^3}+\frac{\Delta x^4}{96}\frac{\partial^4 P}{\partial x^4}+\mathcal{O}\Delta x^5 \\
P(x+\frac{3\Delta x}{2}) &\approx P(x)+\frac{3\Delta x}{2}\frac{\partial P}{\partial x}+\frac{9\Delta x^2}{8}\frac{\partial^2 P}{\partial x^2}+\frac{27\Delta x^3}{24}\frac{\partial^3 P}{\partial x^3}+\frac{81\Delta x^4}{96}\frac{\partial^4 P}{\partial x^4}+\mathcal{O}\Delta x^5 \\
P(x-\frac{3\Delta x}{2}) &\approx P(x)-\frac{3\Delta x}{2}\frac{\partial P}{\partial_x}+\frac{9\Delta x^2}{8}\frac{\partial^2 P}{\partial x^2}-\frac{27\Delta x^3}{24}\frac{\partial^3 P}{\partial x^3}+\frac{81\Delta x^4}{96}\frac{\partial^4 P}{\partial x^4}+\mathcal{O}\Delta x^5
\end{aligned}
$$

Subtracting the expansions of $x-\frac{\Delta x}{2}$ from $x+\frac{\Delta x}{2}$ and subtracting $x-\frac{3\Delta x}{2}$ from $x+\frac{3\Delta x}{2}$ already eliminates the second and fourth order terms (or more general all even-power terms) :

$$
\begin{aligned}
D_1 &= P(x+\frac{\Delta x}{2})-P(x-\frac{\Delta x}{2}) \approx \Delta x\frac{\partial P}{\partial x}+\frac{2\Delta x^3}{24}\frac{\partial^3 P}{\partial x^3}+\mathcal{O}\Delta x^5 \\
D_2 &= P(x+\frac{3\Delta x}{2})-P(x-\frac{3\Delta x}{2}) \approx 3\Delta x\frac{\partial P}{\partial x}+\frac{54\Delta x^3}{24}\frac{\partial^3 P}{\partial x^3}+\mathcal{O}\Delta x^5
\end{aligned}
$$

Using a linear combination of D_1 and D_2, to eliminate the third order term, gives the 4th order approximation:

$$
\frac{27D_1-D_2}{24\Delta x} \approx \frac{\partial P}{\partial x}+\mathcal{O}\Delta x^4 \approx \frac{27(P(x+\frac{\Delta x}{2})-P(x-\frac{\Delta x}{2}))-P(x+\frac{3\Delta x}{2})+P(x-\frac{3\Delta x}{2})}{24\Delta x}+\mathcal{O}\Delta x^4.
\tag{5}
$$

The implemented Finite-Difference codes make use of a staggered grid and is following the grid layout as described in Virieux (1986). In the sections for the specific solutions the staggered grid is explained in detail. The implementation of equation 1 is also called a stencil, since it forms a pattern of four grid point needed to compute the partial derivative at one grid point. To compute the spatial derivative on all grid points the stencil is 'shifted' through the grid.

The medium parameters used in the FD program are

$$(\lambda + 2\mu) = c_p^2 \rho = \frac{1}{\kappa} \tag{6}$$

$$\mu = c_s^2 \rho \tag{7}$$

where ρ is the density of the medium, c_p the P-wave velocity, c_s the S-wave velocity, λ and μ the Lame parameters and κ the compressibility. The program reads the P (and S-wave for elastic modelling) velocity and medium density as gridded input model files. From these files the program calculates the Lame parameters used in the first order equations 1 to calculate the wavefield at next time steps.

1.1 Finite-difference algorithm

To simulate passive seismic measurements we have chosen to use a two-dimensional finite-difference (FD) approach based on the work of Virieux (1986) and Robertsson et al. (1994). The main reason for choosing the finite-difference method is that it runs well on standard X86 and multi-core hardware (including graphical cards) and is easy to implement. For the moment, only the two-dimensional case is implemented to gain experience and be able to run many experiments within a short computation time. For reading input parameters and access files on disk, use is made of the Seismic Unix (SU) parameter interface and SU-segy header format with local IEEE floating point representation for the data. Four different schemes are implemented : acoustic, visco-acoustic, elastic, and visco-elastic. We will not go into all the implementation details and only explain the specific aspects related to the modeling of measurements that can be used to study seismic interferometry (SI). The main difference with other finite-difference codes is the possibility to use band-limited noise signatures positioned at random source positions in the subsurface and model the combined effect of all those sources in only one modeling step. Existing modeling codes are able to model the same result, but are less efficient or less user friendly (more than one program is required to do the modeling off al the passive sources). More details about the used algorithm and the other options within the program can be found in the manual distributed with the code. There are not that many good FD codes available as open source, and we hope that by making the code freely available we would receive requests from users to add new options and keep on expansing and improving the functionality of the code.

Following the flow chart of Figure 1 the algorithm is explained step by step. The program starts by reading in the given parameters and together with default values sets up a modeling experiment. The velocity and density models are read in together with the grid spacing. Using the model grid spacing and the defined time sampling a check is made for the stability and dispersion criteria. The random source positions and signature lengths are computed and all arrays are allocated. The source signatures are calculated in advance and is explained in more detail in section 6.4.

The algorithm contains two loops: the outer loop is for the number of shots and the inner loop for the number of time steps to be modeled for each shot. For seismic interferometry modeling with random source positions the number of shots in the outer loop is set to one, all sources will become active within the inner time loop.

Every time step, the FD kernel is called to update the wavefields and inject source amplitudes, followed by storing of wavefield components on the defined receiver positions and, if requested, a snapshot of the wavefield components is written to disk. The last task within one time step is suppressing reflections from the sides of the model by tapering the edges of the wavefields with an exponentially decaying function. After all time steps are calculated, the stored wavefield components at the receiver positions are written to disk.

In summary FD modeling computes a wavefield (at all gridded x,z positions) at time step $T = t + \Delta t$ given the wavefield at the current time step $T = t$. If during the time stepping of the algorithm a start time of a source is encountered the source amplitude is added to the wavefield at the position of the source.

In the inset of Figure 1 the acoustic FD kernel is sketched in more detail. Inside the kernel, the particle-velocity fields V_x and V_z are updated first. If there are sources active on the particle-velocity fields, these source amplitudes are added to the V_x and V_z fields after the update. This is done for all the defined source positions. Free or rigid boundary conditions are then handled. The pressure field P is updated next and after the update pressure-source amplitudes are added to the pressure field. This last step completes the FD kernel.
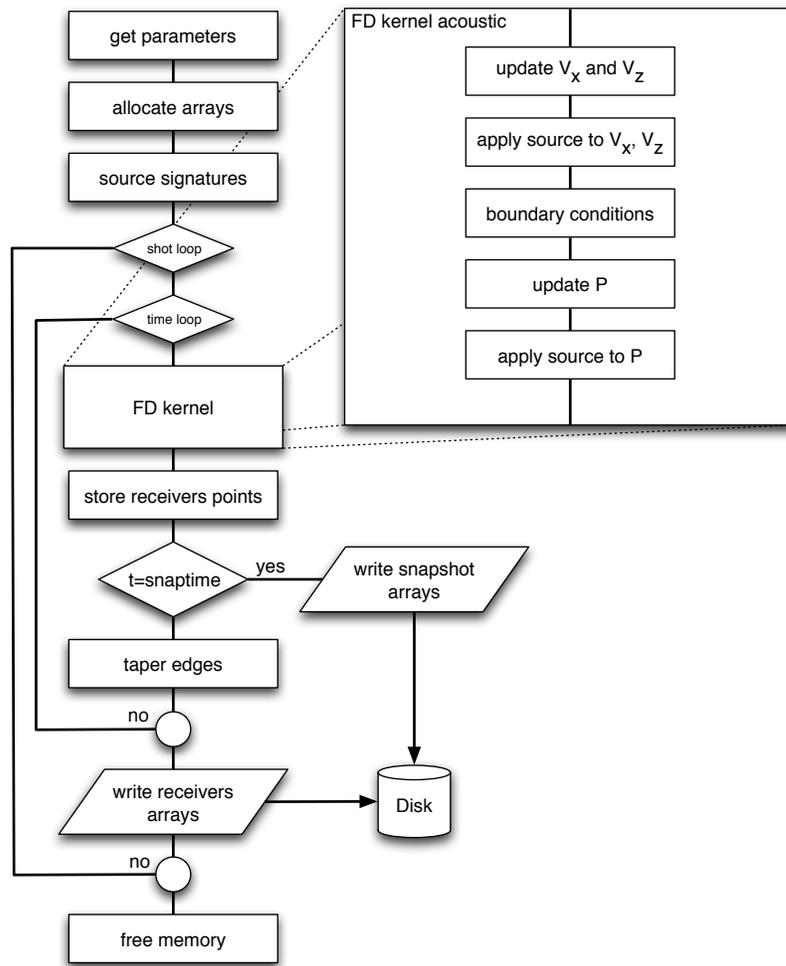
Figure 1: Flow chart of the finite-difference (FD) algorithm. The FD kernel of the acoustic scheme is explained in the onset in more detail. The two decision loops are for the number of shot positions and the number of time steps to be modeled. In the chart, t represents time, V_x and V_z the horizontal and vertical particle-velocity, respectively, and P the acoustic pressure.

The update of the stress fields (P) is done after the updates of the particle-velocity fields, hence the P field is calculated also staggered with time ($+\frac{1}{2}\Delta t$) compared to the particle-velocity fields. Numerical particle velocity (V_x, V_z) are computed at time $(k+\frac{1}{2})\Delta t$, and numerical stress ($\sigma_{zz} = P, \sigma_{xz}, \sigma_{xx}$) at time $(k+1)\Delta t$ are computed explicitly from stress at time $k\Delta t$ and from velocity at time $(k-\frac{1}{2})\Delta t$ (Virieux, 1986).
The kernel operators (stencils) are shown in Figure 2. They are the implementation of the finite-difference approximation of a first order derivative as represented in equation 1. A staggered grid implementation has been used. This means that the grids of the V_x and V_z wavefields are positioned in between the P grid.

a) Kernels to compute update to V_x and V_z b) Kernel to compute update to P
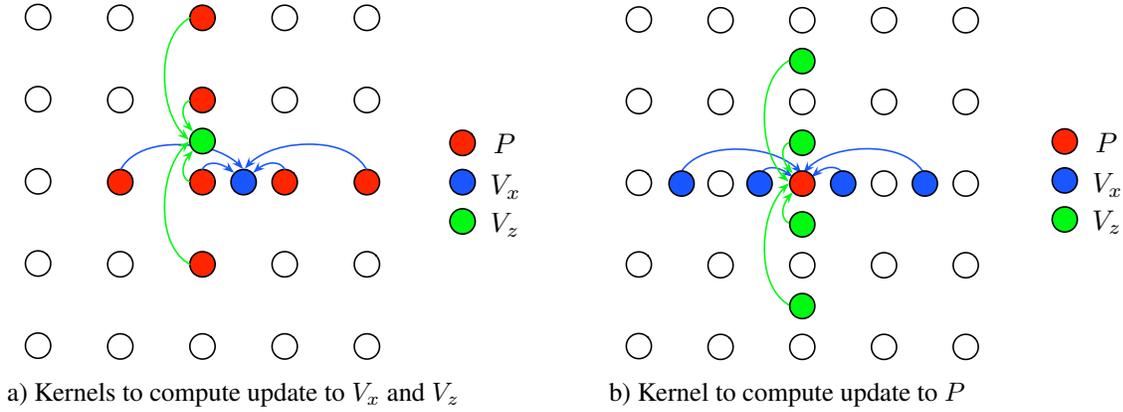
Figure 2: The compute kernels showing the grid points needed to update the V_x and V_z (a) and P (b) wavefields. The wavefields all have a unique grid position. A staggered grid implementation has been used. This means that the grids of the V_x and V_z wavefields are positioned in between the P grid.

1.2 Stability and Dispersion

The first order differential equations are approximated by the finite-difference operators of equations 2 and 3. When explicit time-marching schemes are used for the numerical solution the Courant (Courant et al., 1967) number gives a condition for convergence. The Courant number is used to restrict the time-step in explicit time-marching computer simulations. For example, if a wave is crossing a discrete grid distance (Δx), then the time-step must be less than the time needed for the wave to travel to an adjacent grid point, otherwise the simulation will produce incorrect results. As a corollary, when the grid point separation is reduced, the upper limit for the time step must also decreases. For 4'th order spatial derivatives the Courant number is 0.606 (Sei, 1995) and for stability the discretization must satisfy:

$$\sqrt{\frac{\lambda + 2\mu}{\rho}}\frac{\Delta t}{h} \leq 0.606 \tag{8}$$

$$\tag{9}$$

This approximation requires that

$$\Delta t < \frac{0.606\Delta h}{c_{max}} \tag{10}$$

with $\Delta h = \Delta x = \Delta z$ being the discretization step in the spatial dimensions. If equation 10 is not satisfied unstable results will be calculated if, within a time step Δt, the wavefront has travelled a distance larger than $0.606\Delta x$. This will typically occur at high velocities when $\Delta t c_{max}$ is large. The unstable solution will propagate though the whole model and can end up with large numbers and the out-of-range representation NaN (Not a Number).

Besides unstable solutions wavefield dispersion can also occur. Unfortunately, finite-difference schemes are intrinsically dispersive and there is no fixed grid points per wavelength rule that can be given to avoid dispersion. The widespread rule of thumb "5 points per wavelength" for a (2,4) scheme (Alford et al., 1974) has to be understood in the sense "5 points per wavelength for an average geophysical medium," and for the propagation of a 100 wavelengths through the medium(Sei, 1995). Dispersion for the 2D wave-equation will occur more strongly and clearly visible if the following relation is not obeyed,

$$\Delta h < \frac{c_{min}}{5f_{max}},\tag{11}$$

$$\Delta h < \frac{\lambda_{min}}{5}$$

and will occur at small wavelengths (λ_{min}, low velocities and/or high frequencies). In the case of dispersion, the program will keep on running but will give dispersive waves. Do not confuse numerical dispersion with the physical dispersion of visco-elastic waves discussed later.

a) No dispersion $c_p = 1500, h = 1,$
$\Delta t = 0.0002 \ f_{max} = 260$

b) Dispersion $c_p = 1500, h = 3,$
$\Delta t = 0.0001 \ f_{max} = 260$

c) Dispersion $c_p = 300, h = 1,$
$\Delta t = 0.0002 \ f_{max} = 260$

d) Stability $c_p = 1500, h = 1,$
$\Delta t = 0.0008 \ f_{max} = 260$
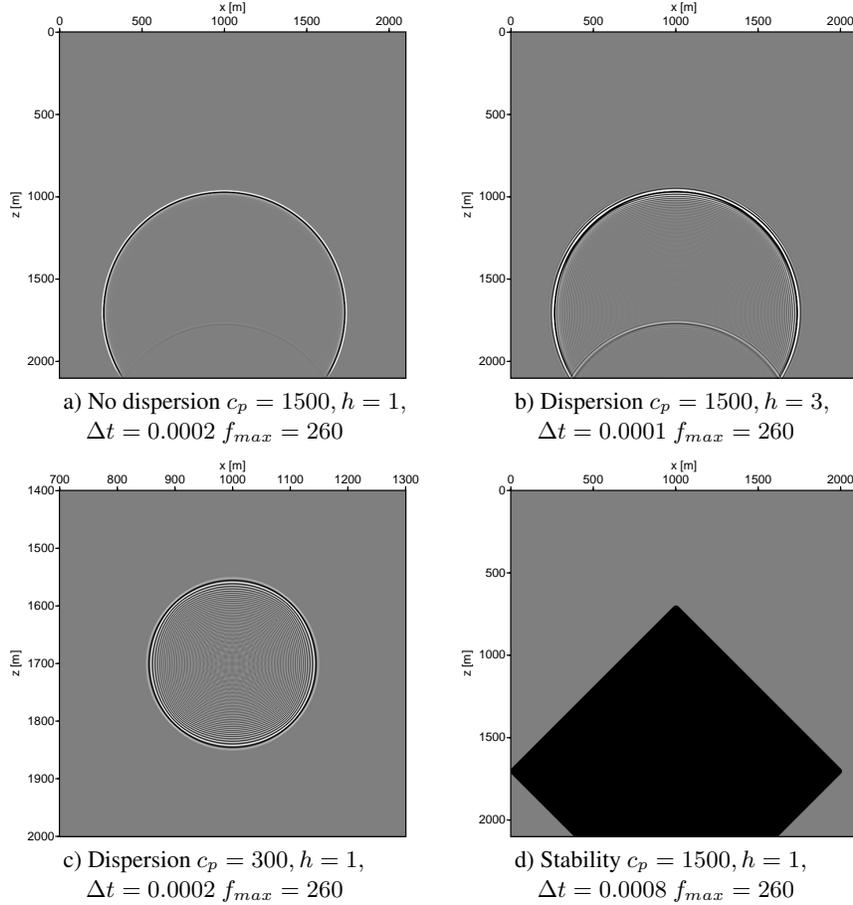
Figure 3: *Snapshots of dispersion (b and c) and unstable (d) schemes. The script* `fdelmodc_stab.scr` *in the demo directory reproduces the pictures.*

Figure 3 shows different snapshots with no dispersion (a), dispersion (b and c) and the results for an unstable scheme (d). Note that before starting the calculating the program checks if the stability and dispersion equations 10 and 11 are satisfied. If they are not satisfied the programs stops with an error message and suggestion how to change the discretization interval or maximum frequency, to get a stable scheme. The dispersion check can be overruled by using the `fmax=` parameter smaller than the actual maximum frequency found in the source wavelet. For a more detailed discussion about stability in finite-difference schemes see Sei (1995), Sei and Symes (1995) Bauer et al. (2008).

Unfortunately, the stability and dispersion criteria shown in equation 10 is not valid for visco-elastic media. See the end of section 5 for some guidelines.

2 Acoustic

The linearized equation of motion (Newton's second law) and equation of deformation (Hook's law) are given by:

$$\frac{\partial V_x}{\partial t} = -\frac{1}{\rho}\frac{\partial P}{\partial x}, \tag{12}$$

$$\frac{\partial V_z}{\partial t} = -\frac{1}{\rho}\frac{\partial P}{\partial z}, \tag{13}$$

$$\frac{\partial P}{\partial t} = -\frac{1}{\kappa}\{\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\}, \tag{14}$$

where V_x, V_z are the particle velocity components in the x and z-direction, respectively, and P the acoustic pressure. In the staggered-grid implementation, ρ_x and V_x, ρ_z and V_z, and $\rho c_p^2 = \frac{1}{\kappa}$ and P are put on the same calculation grid. The computational grid (represented by ρ and c_p) is placed at an offset (one or two grid-points,

8
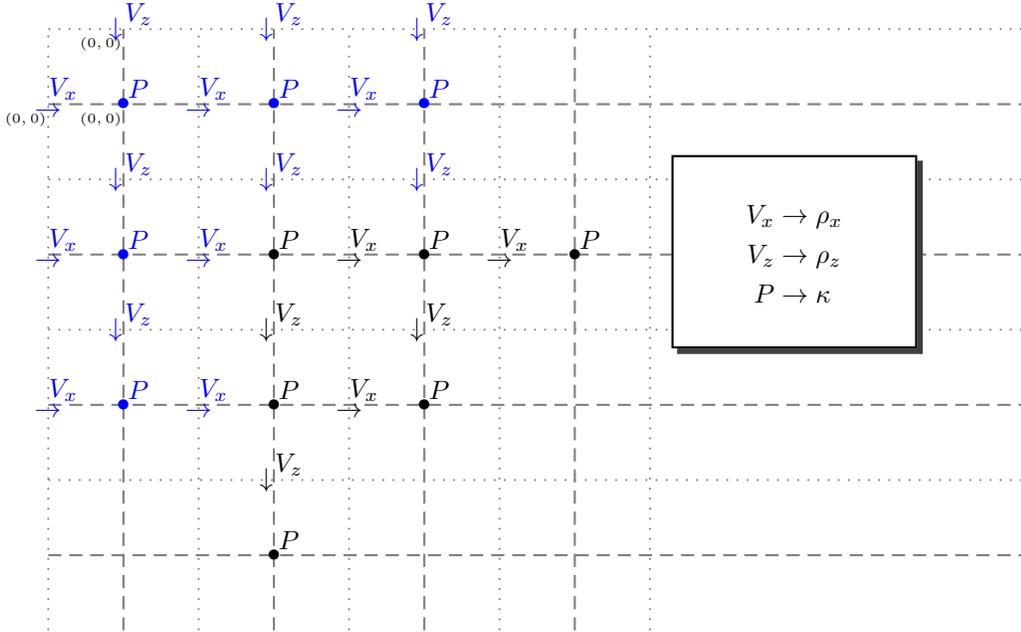
Figure 4: *Acoustic staggered calculation grid for a fourth-order scheme in space. V_z, V_x represent the particle velocity of the wavefield in the z and x direction, respectively, and P represent the acoustic pressure. The blue fields are auxiliary points used to calculate the black field values. Those blue points are not updated and initialized to zero. On all sides of the model a virtual V_x or V_z layer has been added for proper handling of the edges of the model.*

depending of the field component) in the computational grid for efficient handling of the boundaries. Note that compared to equations 1 we have removed the minus sign in the right hand side of 12. This is how the equations are actually implemented in the code. The minus sign is added again in the implementation of the source (so, $-S(t)$ is used). In the other schemes, presented below, the same notation without the minus sign is used.

The pressure/stress wavefields are computed on different time steps than the particle-velocity fields. In the algorithm the first time-step ($it = 0$) computed for the particle-velocity fields uses pressure/stress fields at $it = 0$ resulting in particle-velocity fields at time steps $(it + \frac{1}{2})\Delta t$. The pressure/stress fields are computed at time steps $it\Delta t$ using the particle-velocity fields at time steps $(it + \frac{1}{2})\Delta t$.

For the staggered-grid implementation, shown in Figure 4, every field quantity has a different origin. The origins of the field are chosen in such a way that interpolation of one field grid to another field grid can be done in a straightforward way (see also section 6.6.3). The derivative operators need two points on each sides of their centre to calculate the derivative at the centre. By offsetting the grid, the extra points needed to calculate the derivative at the boundaries of the model are added. These extra layers, needed at the edges of the model, are also taken into account in the choice of the origin. The origins of the medium parameters (and the different fields) are defined according to the following mapping:

$$[z, x]$$
$$\rho_x[1, 2] \leftarrow 0.5 * (\rho[0, 0] + \rho[0, 1])$$
$$\rho_z[2, 1] \leftarrow 0.5 * (\rho[0, 0] + \rho[1, 0])$$
$$\kappa[1, 1] \leftarrow c_p^2[0, 0]\rho[0, 0].$$

$$(15)$$

Note that the choice for the origin is just a choice for convenience and nothing else.

In the code section below the `io**` variables define the origin-offsets used in the calculations.

2.1 Staggered scheme

```
c1 = 9.0/8.0;
c2 = -1.0/24.0;
```

```
/* Vx: rox */
ioXx=2;
ioXz=ioXx-1;
/* Vz: roz */
ioZz=2;
ioZx=ioZz-1;
/* P, Txx, Tzz: lam, l2m */
ioPx=1;
ioPz=ioPx;
/* Txz: muu */
ioTx=2;
ioTz=ioTx;

rox = 1/rho_x * dt/dx
roz = 1/rho_z * dt/dx
l2m = cp*cp*rho * dt/dx

/* calculate vx for all grid points except on the virtual boundary*/
for (ix=ioXx; ix<nx+1; ix++) {
    for (iz=ioXz; iz<nz+1; iz++) {
        vx[ix*n1+iz] -= rox[ix*n1+iz]*(
                    c1*(p[ix*n1+iz]     - p[(ix-1)*n1+iz]) +
                    c2*(p[(ix+1)*n1+iz] - p[(ix-2)*n1+iz]));
    }
}


/* calculate vz for all grid points except on the virtual boundary */
for (ix=ioZx; ix<nx+1; ix++) {
    for (iz=ioZz; iz<nz+1; iz++) {
        vz[ix*n1+iz] -= roz[ix*n1+iz]*(
                    c1*(p[ix*n1+iz]   - p[ix*n1+iz-1]) +
                    c2*(p[ix*n1+iz+1] - p[ix*n1+iz-2]));
    }
}



/* calculate p/tzz for all grid points except on the virtual boundary */
for (ix=ioPx; ix<nx+1; ix++) {
    for (iz=ioPz; iz<nz+1; iz++) {
        p[ix*n1+iz] -= l2m[ix*n1+iz]*(
                    c1*(vx[(ix+1)*n1+iz] - vx[ix*n1+iz]) +
                    c2*(vx[(ix+2)*n1+iz] - vx[(ix-1)*n1+iz]) +
                    c1*(vz[ix*n1+iz+1]   - vz[ix*n1+iz]) +
                    c2*(vz[ix*n1+iz+2]   - vz[ix*n1+iz-1]));
    }
}
```

3 Visco-Acoustic

For a visco-acoustic medium the linearized equation of motion (Newton's second law) and equation of deformation (Hook's law) are :

$$\frac{\partial V_x}{\partial t} = -\frac{1}{\rho}\frac{\partial P}{\partial x} \tag{16}$$

$$\frac{\partial V_z}{\partial t} = -\frac{1}{\rho}\frac{\partial P}{\partial z} \tag{17}$$

$$\frac{\partial P}{\partial t} = -\frac{1}{\kappa}\frac{\tau_\varepsilon^p}{\tau_\sigma}\{\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\} + r_p \tag{18}$$

$$\frac{\partial r_p}{\partial t} = -\frac{1}{\tau_\sigma}\left(r_p + (\frac{\tau_\varepsilon^p}{\tau_\sigma} - 1)(\frac{1}{\kappa})\{\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\}\right) \tag{19}$$

For the attenuation implementation a leap-frog scheme in time is used and shown in the implementation below. The so-called memory variable r_p (Robertsson et al., 1994) are introduced for the relaxation mechanism.

```
/* calculate p/tzz for all grid points except on the virtual boundary */
for (ix=ioPx; ix<nx+1; ix++) {
    for (iz=ioPz; iz<nz+1; iz++) {
        dxvx[iz] = c1*(vx[(ix+1)*n1+iz] - vx[ix*n1+iz]) +
                   c2*(vx[(ix+2)*n1+iz] - vx[(ix-1)*n1+iz]);
    }
    for (iz=ioPz; iz<nz+1; iz++) {
        dzvz[iz] = c1*(vz[ix*n1+iz+1]   - vz[ix*n1+iz]) +
                   c2*(vz[ix*n1+iz+2]   - vz[ix*n1+iz-1]);
    }

    /* help variables to let the compiler vectorize the loops */
    for (iz=ioPz; iz<nz+1; iz++) {
        Tpp     = tep[ix*n1+iz]*tss[ix*n1+iz];
        Tlm[iz] = (1.0-Tpp)*tss[ix*n1+iz]*l2m[ix*n1+iz]*0.5;
        Tlp[iz] = l2m[ix*n1+iz]*Tpp;
    }
    for (iz=ioPz; iz<nz+1; iz++) {
        Tt1[iz] = 1.0/(ddt+0.5*tss[ix*n1+iz]);
        Tt2[iz] = ddt-0.5*tss[ix*n1+iz];
    }

    /* the update with the relaxation correction */
    for (iz=ioPz; iz<nz+1; iz++) {
        p[ix*n1+iz] -= Tlp[iz]*(dzvz[iz]+dxvx[iz]) + q[ix*n1+iz];
    }
    for (iz=ioPz; iz<nz+1; iz++) {
        q[ix*n1+iz] = (Tt2[iz]*q[ix*n1+iz] + Tlm[iz]*(dxvx[iz]+dzvz[iz]))*Tt1[iz];
        p[ix*n1+iz] += q[ix*n1+iz];
    }
}
```

The relaxation parameters $\tau_\varepsilon^p, \tau_\sigma$ are defined in the program indirectly by defining the Quality factor (Q). This Q-factor can be defined in the program by setting the parameter Qp= for a constant-Q medium,or by using a gridded file file_qp=, which defines a different Q-factor for every grid point. The Q-factors are transformed inside the
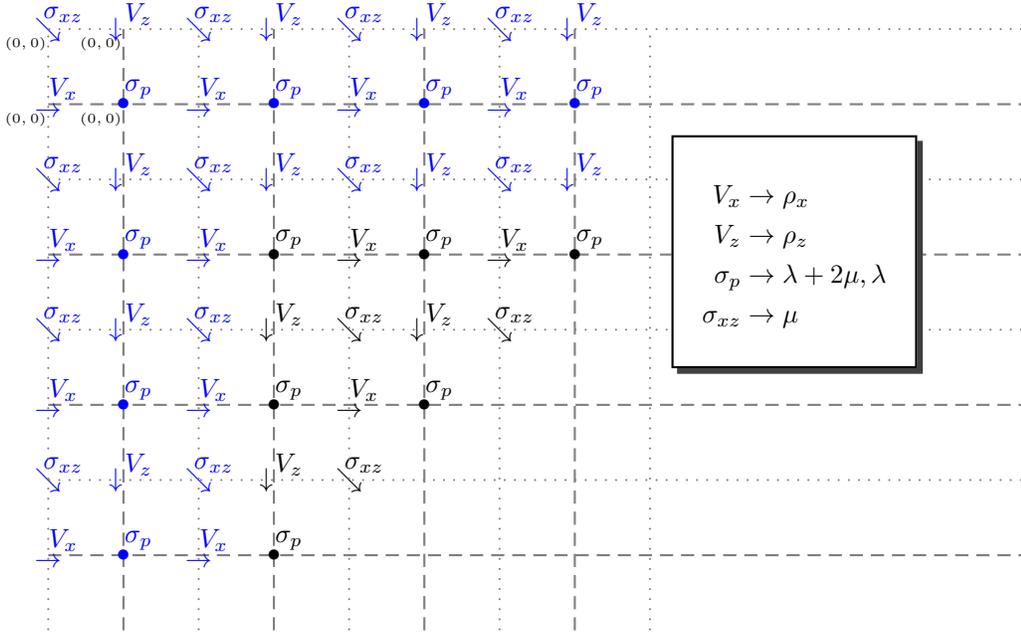
Figure 5: *Elastic staggered calculation grid for a fourth-order scheme in space. V_z, V_x represent the particle velocity of the wavefield in the z and x direction, respectively, and $\sigma_p(\sigma_{xx}$ or $\sigma_{zz}), \sigma_{xz}$ represent the stress fields. The blue fields are auxiliary points used to calculate the black field values. Those blue points are not updated and initialized to zero. On all sides of the model a virtual V_x, σ_{xz} or V_z, σ_{xz} layer has been added for proper handling of the edges of the model.*

program to the relaxation parameters (used in the numerical scheme) by using (Robertsson et al., 1994):

$$Q = \frac{t_s \left(1 + w^2 t_s t_e\right)}{(t_e - t_s)\, w t_s} \tag{20}$$

$$\tau_\sigma = \frac{\sqrt{1.0 + \frac{1.0}{Q_p^2}} - \frac{1.0}{Q_p}}{f_w} \tag{21}$$

$$\tau_\varepsilon^p = \frac{1.0}{f_w^2 \tau_\sigma} \tag{22}$$

$$\tau_\varepsilon^s = \frac{1.0 + f_w Q_s \tau_\sigma}{f_w Q_s - f_w^2 \tau_\sigma} \tag{23}$$

where f_w is the central frequency (of the used wavelet) given by parameter fw=.
The relaxation parameters are defined by the following damping model:

$$M(\omega) = k_0 \left(1 - L \sum_{l=1}^{L} \frac{1 + j\omega t_{e,l}}{1 + j\omega t_{s,l}}\right) \tag{24}$$

$$Q = \frac{\Re\{M(\omega)\}}{\Im\{M(\omega)\}} \tag{25}$$

TODO: explain the physical mechanism of the used damping model (mass-spring configuration).

4 Elastic

Linearized equation of motion (Newton's second law) and equation of deformation (Hook's law) are used:

$$\frac{\partial V_x}{\partial t} = -\frac{1}{\rho}\{\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xz}}{\partial z}\} \tag{26}$$

$$\frac{\partial V_z}{\partial t} = -\frac{1}{\rho}\{\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{zz}}{\partial z}\} \tag{27}$$

$$\frac{\partial \sigma_{xx}}{\partial t} = -\{\frac{1}{\kappa}\frac{\partial V_x}{\partial x} + \lambda\frac{\partial V_z}{\partial z}\} \tag{28}$$

$$\frac{\partial \sigma_{zz}}{\partial t} = -\{\frac{1}{\kappa}\frac{\partial V_z}{\partial z} + \lambda\frac{\partial V_x}{\partial x}\} \tag{29}$$

$$\frac{\partial \sigma_{xz}}{\partial t} = -\mu\{\frac{\partial V_x}{\partial z} + \frac{\partial V_z}{\partial x}\} \tag{30}$$

where σ_{ij} denotes the ijth component of the symmetric stress tensor Virieux (1986).

The derivative operators need two points on each side of their centre to calculate the derivative at the centre. By offsetting the grid, the extra points needed to calculate the derivative at the boundaries of the model are added. These extra layers needed at the edges of the model are also taken into account in the choice of the origin. The origins are defined according to the following mapping:

$$[z, x]$$
$$\rho_x[1, 2] \leftarrow 0.5 * (\rho[0, 0] + \rho[0, 1])$$
$$\rho_z[2, 1] \leftarrow 0.5 * (\rho[0, 0] + \rho[1, 0])$$
$$\kappa[1, 1] \leftarrow c_p^2[0, 0]\rho[0, 0]$$
$$\mu[2, 2] \leftarrow c_s^2[0, 0]\rho[0, 0]$$
$$\lambda[1, 1] \leftarrow c_p^2[0, 0]\rho[0, 0] - 2c_p^2[0, 0]\rho[0, 0].$$

```
/* Vx: rox */
ioXx=mod.iorder/2;
ioXz=ioXx-1;
/* Vz: roz */
ioZz=mod.iorder/2;
ioZx=ioZz-1;
/* P, Txx, Tzz: lam, l2m */
ioPx=mod.iorder/2-1;
ioPz=ioPx;
/* Txz: muu */
ioTx=mod.iorder/2;
ioTz=ioTx;

/* calculate vx for all grid points except on the virtual boundary*/
for (ix=ioXx; ix<nx+1; ix++) {
    for (iz=ioXz; iz<nz+1; iz++) {
        vx[ix*n1+iz] -= rox[ix*n1+iz]*(
                    c1*(txx[ix*n1+iz]     - txx[(ix-1)*n1+iz] +
                        txz[ix*n1+iz+1]   - txz[ix*n1+iz])    +
                    c2*(txx[(ix+1)*n1+iz] - txx[(ix-2)*n1+iz] +
                        txz[ix*n1+iz+2]   - txz[ix*n1+iz-1])  );
    }
}


/* calculate vz for all grid points except on the virtual boundary */
for (ix=ioZx; ix<nx+1; ix++) {
    for (iz=ioZz; iz<nz+1; iz++) {
```

```
            vz[ix*n1+iz] -= roz[ix*n1+iz]*(
                        c1*(tzz[ix*n1+iz]      - tzz[ix*n1+iz-1] +
                            txz[(ix+1)*n1+iz] - txz[ix*n1+iz])   +
                        c2*(tzz[ix*n1+iz+1]    - tzz[ix*n1+iz-2] +
                            txz[(ix+2)*n1+iz] - txz[(ix-1)*n1+iz])  );
    }
}

/* calculate Txx/tzz for all grid points except on the virtual boundary */
for (ix=ioPx; ix<nx+1; ix++) {
    for (iz=ioPz; iz<nz+1; iz++) {
        dvvx[iz] = c1*(vx[(ix+1)*n1+iz] - vx[ix*n1+iz]) +
                   c2*(vx[(ix+2)*n1+iz] - vx[(ix-1)*n1+iz]);
    }
    for (iz=ioPz; iz<nz+1; iz++) {
        dvvz[iz] = c1*(vz[ix*n1+iz+1]   - vz[ix*n1+iz]) +
                   c2*(vz[ix*n1+iz+2]   - vz[ix*n1+iz-1]);
    }
    for (iz=ioPz; iz<nz+1; iz++) {
        txx[ix*n1+iz] -= l2m[ix*n1+iz]*dvvx[iz] + lam[ix*n1+iz]*dvvz[iz];
        tzz[ix*n1+iz] -= l2m[ix*n1+iz]*dvvz[iz] + lam[ix*n1+iz]*dvvx[iz];
    }
}

/* calculate Txz for all grid points except on the virtual boundary */
for (ix=ioTx; ix<nx+1; ix++) {
    for (iz=ioTz; iz<nz+1; iz++) {
        txz[ix*n1+iz] -= mul[ix*n1+iz]*(
                    c1*(vx[ix*n1+iz]      - vx[ix*n1+iz-1] +
                        vz[ix*n1+iz]      - vz[(ix-1)*n1+iz]) +
                    c2*(vx[ix*n1+iz+1]    - vx[ix*n1+iz-2] +
                        vz[(ix+1)*n1+iz] - vz[(ix-2)*n1+iz]) );
    }
}
```

To handle a solid fluid interface an extra layer is introduced between the interfaces. This technique is described by van Vossen et al. (2002).

5 Visco-Elastic

Linearized equation of motion (Newton's second law) and equation of deformation (Hook's law) used are:

$$\frac{\partial V_x}{\partial t} = -\frac{1}{\rho}\{\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xz}}{\partial z}\} \tag{31}$$

$$\frac{\partial V_z}{\partial t} = -\frac{1}{\rho}\{\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{zz}}{\partial z}\} \tag{32}$$

$$\frac{\partial \sigma_{xx}}{\partial t} = -\frac{1}{\kappa}\frac{\tau_\varepsilon^p}{\tau_\sigma}\{\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\} - 2\mu\frac{\tau_\varepsilon^s}{\tau_\sigma}\frac{\partial V_z}{\partial z} + r_{xx} \tag{33}$$

$$\frac{\partial \sigma_{zz}}{\partial t} = -\frac{1}{\kappa}\frac{\tau_\varepsilon^p}{\tau_\sigma}\{\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\} - 2\mu\frac{\tau_\varepsilon^s}{\tau_\sigma}\frac{\partial V_x}{\partial x} + r_{zz} \tag{34}$$

$$\frac{\partial \sigma_{xz}}{\partial t} = -\mu\frac{\tau_\varepsilon^s}{\tau_\sigma}\{\frac{\partial V_x}{\partial z} + \frac{\partial V_z}{\partial x}\} + r_{xz} \tag{35}$$

$$\frac{\partial r_{xx}}{\partial t} = -\frac{1}{\tau_\sigma}\left(r_{xx} + (\frac{\tau_\varepsilon^p}{\tau_\sigma}-1)(\frac{1}{\kappa})\{\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\} - (\frac{\tau_\varepsilon^s}{\tau_\sigma}-1)2\mu\frac{\partial V_z}{\partial z}\right) \tag{36}$$

$$\frac{\partial r_{zz}}{\partial t} = -\frac{1}{\tau_\sigma}\left(r_{zz} + (\frac{\tau_\varepsilon^p}{\tau_\sigma}-1)(\frac{1}{\kappa})\{\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\} - (\frac{\tau_\varepsilon^s}{\tau_\sigma}-1)2\mu\frac{\partial V_x}{\partial x}\right) \tag{37}$$

$$\frac{\partial r_{xz}}{\partial t} = -\frac{1}{\tau_\sigma}\left(r_{xz} + (\frac{\tau_\varepsilon^s}{\tau_\sigma}-1)\mu\{\frac{\partial V_x}{\partial z} + \frac{\partial V_x}{\partial z}\}\right) \tag{38}$$

More details about visco-elastic FD modelling can be found in Robertsson et al. (1994), Saenger and Bohlen (2004), and Bohlen (2002).

The relaxation parameters $\tau_\varepsilon^p, \tau_\varepsilon^s, \tau_\sigma$ are defined in the program indirectly by defining the Quality factor (Q). This Q-factor can be defined in the program by setting the parameter `Qp= Qs=` for a constant-Q medium, or by using a gridded file `file_qp= file_qs=` which defines a different Q-factor for every grid point. The Q-factors are in the program transformed into the relaxation parameters (used in the numerical scheme) by using Robertsson et al. (1994):

$$Q = \frac{t_s\left(1 + w^2 t_s t_e\right)}{(t_e - t_s)\,w t_s} \tag{39}$$

$$\tau_\sigma = \frac{\sqrt{1.0 + \frac{1.0}{Q_p^2}} - \frac{1.0}{Q_p}}{f_w} \tag{40}$$

$$\tau_\varepsilon^p = \frac{1.0}{f_w^2 \tau_\sigma} \tag{41}$$

$$\tau_\varepsilon^s = \frac{1.0 + f_w Q_s \tau_\sigma}{f_w Q_s - f_w^2 \tau_\sigma} \tag{42}$$

where f_w is the central frequency (of the used wavelet) given by parameter `fw=`.

The relaxation parameters are defined by the following damping model:

$$M(\omega) = k_0 \left(1 - L\sum_{l=1}^{L}\frac{1 + j\omega t_{e,l}}{1 + j\omega t_{s,l}}\right) \tag{43}$$

$$Q = \frac{\Re\{M(\omega)\}}{\Im\{M(\omega)\}} \tag{44}$$

Unfortunately, the stability calculations within the program are not always valid for visco-elastic media. There is no general rule of thumb for visco-elastic media to calculate a stability criterion. If in a modeling experiment, with a chosen Qp and Qs, the program is not stable, the advise is to increase the smallest Q-factor (for example from 20 to 30), or use `ischeme=3`, and see if that gives a stable modeling result. If you get a stable result by increasing Q (or `ischeme=3`) and you want to use the lower Q, value you have to make the dx (and possibly dt) smaller to get stable answers for that Q value as well.

6   Parameters in program fdelmodc

The self-doc of the program is shown by typing `fdelmodc` on the command line without any arguments. You will then see the following exhaustive list of parameters:

```
fdelmodc - elastic acoustic finite difference wavefield modeling

IO PARAMETERS:
  file_cp= .......... P (cp) velocity file
  file_cs= .......... S (cs) velocity file
  file_den= ........ density (ro) file
  file_src= ........ file with source signature
  file_rcv=recv.su .. base name for receiver files
  file_snap=snap.su . base name for snapshot files
  file_beam=beam.su . base name for beam fields
  dx= .............. read from model file: if dx==0 then dx= can be used to set it
  dz= .............. read from model file: if dz==0 then dz= can be used to set it
  dt= .............. read from file_src: if dt==0 then dt= can be used to set it

OPTIONAL PARAMETERS:
  ischeme=3 ........ 1=acoustic, 2=visco-acoustic 3=elastic, 4=visco-elastic
  tmod=(nt-1)*dt .... total registration time (nt from file_src)
  ntaper=0 .......... length of taper in points at edges of model
  npml=35 ........... length of PML layer in points at edges of model
  R=1e-4 ............ the theoretical reflection coefficient at PML boundary
  m=2.0 ............. scaling order of the PML sigma function
  tapfact=0.30 ...... taper strength: larger value gets stronger taper
  For the 4 boundaries the options are:  1=free 2=pml 3=rigid 4=taper
  top=1 ............. type of boundary on top edge of model
  left=4 ............ type of boundary on left edge of model
  right=4 ........... type of boundary on right edge of model
  bottom=4 ......... type of boundary on bottom edge of model
  grid_dir=0 ....... direction of time modeling (1=reverse time)
  Qp=15 ............ global Q-value for P-waves in visco-elastic (ischeme=2,4)
  file_qp= ......... model file Qp values as function of depth
  Qs=Qp ............ global Q-value for S-waves in visco-elastic (ischeme=4)
  file_qs= ......... model file Qs values as function of depth
  fw=0.5*fmax ....... central frequency for which the Q's are used
  sinkdepth=0 ...... receiver grid points below topography (defined bij cp=0.0)
  sinkdepth_src=0 ... source grid points below topography (defined bij cp=0.0)
  sinkvel=0 ........ use velocity of first receiver to sink through to next layer
  beam=0 ........... calculate energy beam of wavefield in model
  disable_check=0 ... disable stabilty and dispersion check and continue modeling
  verbose=0 ........ silent mode; =1: display info

SHOT AND GENERAL SOURCE DEFINITION:
  src_type=1 ....... 1=P 2=Txz 3=Tzz 4=Txx 5=S-pot 6=Fx 7=Fz 8=P-pot
  src_orient=1 ...... orientation of the source
                   - 1=monopole
                   - 2=dipole +/- vertical oriented
                   - 3=dipole - + horizontal oriented
  xsrc=middle ....... x-position of (first) shot
  zsrc=zmin ........ z-position of (first) shot
  nshot=1 .......... number of shots to model
  dxshot=dx ........ if nshot > 1: x-shift in shot locations
  dzshot=0 ......... if nshot > 1: z-shift in shot locations
  xsrca= ........... defines source array x-positions
```

```
    zsrca= ............ defines source array z-positions
    wav_random=1 ...... 1 generates (band limited by fmax) noise signatures
    fmax=from_src ..... maximum frequency in wavelet
    src_multiwav=0 .... use traces in file_src as areal source
    src_at_rcv=1 ...... inject wavefield at receiver coordinates (1), inject at source (0)
    src_injectionrate=0 set to 1 to use injection rate source

PLANE WAVE SOURCE DEFINITION:
  plane_wave=0 ...... model plane wave with nsrc= sources
  nsrc=1 ............ number of sources per (plane-wave) shot
  src_angle=0 ....... angle of plane source array
  src_velo=1500 ..... velocity to use in src_angle definition
  src_window=0 ...... length of taper at edges of source array

RANDOM SOURCE DEFINITION FOR SEISMIC INTERFEROMTERY:
  src_random=0 ...... 1 enables nsrc random sources positions in one modeling
  nsrc=1 ............ number of sources to use for one shot
  xsrc1=0 ........... left bound for x-position of sources
  xsrc2=0 ........... right bound for x-position of sources
  zsrc1=0 ........... left bound for z-position of sources
  zsrc2=0 ........... right bound for z-position of sources
  tsrc1=0.0 ........ begin time interval for random sources being triggered
  tsrc2=tmod ....... end time interval for random sources being triggered
  tactive=tsrc2 ..... end time for random sources being active
  tlength=tsrc2-tsrc1 average duration of random source signal
  length_random=1 ... duration of source is rand*tlength
  amplitude=0 ....... distribution of source amplitudes
  distribution=0 .... random function for amplitude and tlength 0=flat 1=Gaussian
  seed=10 ........... seed for start of random sequence

SNAP SHOT SELECTION:
  tsnap1=0.1 ........ first snapshot time (s)
  tsnap2=0.0 ........ last snapshot time (s)
  dtsnap=0.1 ........ snapshot time interval (s)
  dxsnap=dx ......... sampling in snapshot in x-direction
  xsnap1=0 .......... first x-position for snapshots area
  xsnap2=0 .......... last x-position for snapshot area
  dzsnap=dz ......... sampling in snapshot in z-direction
  zsnap1=0 .......... first z-position for snapshots area
  zsnap2=0 .......... last z-position for snapshot area
  snapwithbnd=0 ..... write snapshots with absorbing boundaries
  sna_type_p=1 ...... p registration _sp
  sna_type_vz=1 ..... Vz registration _svz
  sna_type_vx=0 ..... Vx registration _svx
  sna_type_txx=0 .... Txx registration _stxx
  sna_type_tzz=0 .... Tzz registration _stzz
  sna_type_txz=0 .... Txz registration _stxz
  sna_type_pp=0 ..... P (divergence) registration _sP
  sna_type_ss=0 ..... S (curl) registration _sS
  sna_vxvztime=0 .... registration of vx/vx times
                     The fd scheme is also staggered in time.
                     Time at which vx/vz snapshots are written:
                   - 0=previous vx/vz relative to txx/tzz/txz at time t
                   - 1=next    vx/vz relative to txx/tzz/txz at time t

RECEIVER SELECTION:
  xrcv1=xmin ........ first x-position of linear receiver array(s)
```

```
xrcv2=xmax ........ last x-position of linear receiver array(s)
dxrcv=dx .......... x-position increment of receivers in linear array(s)
zrcv1=zmin ........ first z-position of linear receiver array(s)
zrcv2=zrcv1 ....... last z-position of linear receiver array(s)
dzrcv=0.0 ......... z-position increment of receivers in linear array(s)
dtrcv=.004 ........ desired sampling in receiver data (seconds)
xrcva= ............ defines receiver array x-positions
zrcva= ............ defines receiver array z-positions
rrcv= ............. radius for receivers on a circle
arcv= ............. vertical arc-lenght for receivers on a ellipse (rrcv=horizontal)
oxrcv=0.0 ......... x-center position of circle
ozrcv=0.0 ......... z-center position of circle
dphi=2 ............ angle between receivers on circle
rcv_txt=........... text file with receiver coordinates. Col 1: x, Col. 2: z
rec_ntsam=nt ...... maximum number of time samples in file_rcv files
rec_delay=0 ....... time in seconds to start recording: recorded time = tmod - rec_del
rec_type_p=1 ...... p registration _rp
rec_type_vz=1 ..... Vz registration _rvz
rec_type_vx=0 ..... Vx registration _rvx
rec_type_txx=0 .... Txx registration _rtxx
rec_type_tzz=0 .... Tzz registration _rtzz
rec_type_txz=0 .... Txz registration _rtxz
rec_type_pp=0 ..... P (divergence) registration _rP
rec_type_ss=0 ..... S (curl) registration _rS
rec_type_ud=0 ..... decomposition in up and downgoing waves _ru, _rd
kangle= ........... maximum wavenumber angle for decomposition
rec_int_vx=0  ..... interpolation of Vx receivers
                    - 0=Vx->Vx (no interpolation)
                    - 1=Vx->Vz
                    - 2=Vx->Txx/Tzz(P)
                    - 3=Vx->receiver position
rec_int_vz=0 ...... interpolation of Vz receivers
                    - 0=Vz->Vz (no interpolation)
                    - 1=Vz->Vx
                    - 2=Vz->Txx/Tzz(P)
                    - 3=Vz->receiver position
rec_int_p=0  ...... interpolation of P/Tzz receivers
                    - 0=P->P (no interpolation)
                    - 1=P->Vz
                    - 2=P->Vx
                    - 3=P->receiver position

 NOTES: For viscoelastic media dispersion and stability are not always
 guaranteed by the calculated criteria, especially for Q values smaller than 13

    Jan Thorbecke 2011
    TU Delft
    E-mail: janth@xs4all.nl
    2015  Contributions from Max Holicki
```

If you are not considering doing special things, the default values are most of the times sufficient and only a few parameters have to be changed from their default values. For all types of FD modeling experiments, the medium parameters must be given. The medium parameters describe the discretized medium through which the modelling is carried out. The source wavelet must also be given. Besides that no other parameters are needed and the program will start modelling with a source positioned at the top middle of the model (z), with receivers placed at the top with a distance equal to the grid distance. This minimum parameter set is:

```
fdelmodc file_cp=filecp.su file_cs=filecs.su file_den=filero.su \
```

```
file_src=wavelet.su
```

In the next subsections all the parameters will be described in more detail and guidelines will be given how to use them.

6.1 Modelling parameters

The `ischeme=` selects the kind of finite-difference scheme to be used. Currently there are four options:

1. acoustic, see section 2

2. visco-acoustic, see section 3

3. elastic, see section 4

4. visco-elastic, see section 5

For visco-acoustic (elastic) media extra options are: `Qp=` (and `Qs=`) for selecting an overall Q factor for all layers. This Q value is defined for a frequency at `fw=`, other frequencies will have slightly different Q values. It is also possible to define a Q value for every grid point in the medium. These arrays must be stored in SU files, have the same dimensions as the files of the gridded medium parameters. The program parameters `file_qp=` (and `file_qs= for elastic media`) will read these Q defined values. You can use makemod to define a model and use one of medium parameters as Q value. For example you can use the cs layer parameter to define a different Q-factor in each layer. You can make you own gridded Q definition in for example Matlab; make sure that this gridded file has same size as the file_cp/file_ro grids.

6.2 Medium parameters

The parameters `file_cp, file_cs, file_den` represent the filenames of the gridded model files in SU format. The fastest dimension (n_1, number of samples per trace, z) in the file represents depth and the second dimension (n_2, number of traces, x) represents the lateral position. The distance between the grid points has to be the same in the z and x position. The grid distance is read from the headers `d1, d2` in the model files. The origin of the model is read from the headers `f1, f2`, where `f1` is the depth of the first sample and `f2` is the lateral position of the first trace. If those headers are not set then the user can define the sampling distance by using the parameters `dx= dz=`. The output files (receivers and snapshots) will also contains the lateral coordinates in the `gx` headers. The gridded model files can be generated by the, also provided, program `makemod` in the `utils` directory.

Together with the minimum and maximum velocities in the model files, the spatial- and time-sampling the stability of the solution can be calculated (see section 1.2).

The dimension of the velocity files is $[m/s]$ and for the density it is $[kg/m^3]$. The unit for the length direction can also be feet (or any other length measurement), as long as the same length unit is used on the distance and the velocity of the medium.

If you want to make a region where no waves propagate; define the (P and or S) velocity to zero, but not the density. Otherwise fdelmodc will give an error message:

```
Warning in fdelmodc: Zero density for trace=0 sample=10;
Error in fdelmodc: ERROR zero density is not a valid value, program exit
```

In the algorithms the reciprocal value of the density is used. To avoid checking zero densities for each loop (and therefore make the code perform slower) zero densities are not allowed.

6.3 Boundaries

There are four boundary types used in the FD schemes. The boundary type is selected with the parameters `left= right= top= bottom=` and are identified with a number . The default values of these parameters are; free surface for the top (1) and tapered (4) for the other three boundaries. The different boundary types are:

- 'Absorbing' tapered boundaries (4)

  One of the most important boundary type is the absorbing boundary. This type of boundary is used to avoid reflections from the sides of the model. The boundaries in a numerical model are in most cases not physical boundaries, but artificial boundaries introduced to limit the size of the model. Reflections coming from these boundaries are artificial and must therefore be suppressed. There are many possible implementations to absorb these artificial reflections. In the program, the most simple absorbing boundary condition is implemented: a taper on the V_x and V_z fields. It is difficult to give guidelines how many grid points the taper length should be to suppress the side reflections. The wave field is gradually tapered over a specified range of grid points (window length `ntaper`). The default setting for the taper length is: $4 * ((cp_max/fmax)/dx)$, 4 wavelengths. Depending on the size of the grid a window length of 40,80 grid points might be sufficient. You may alter these, if you like, in order to increase or decrease the amount of tapering. The larger the window length, the better the absorption, but the longer the modeling will take.

a) no taper   b) 200 points taper
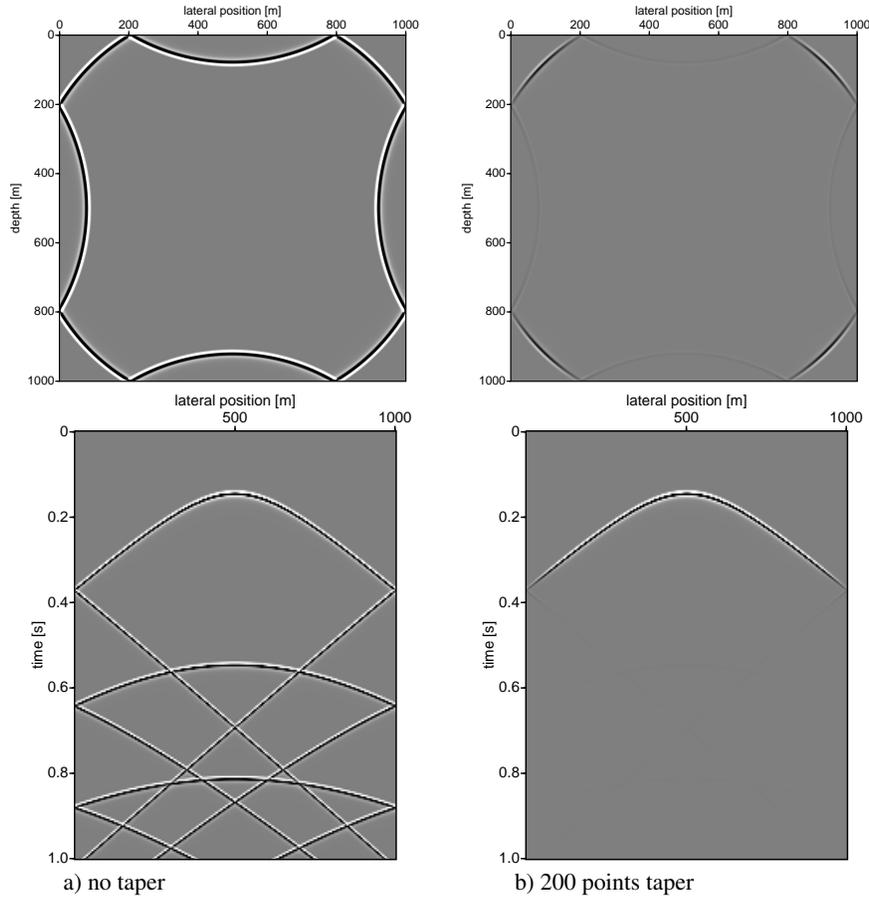
Figure 6: *Snapshots and receiver recording in homogeneous medium with and without taper. The receivers are placed at 300 m depth and the source is positioned in the middle of the model at (500,500). The grid distance is 1 meter. The script* `fdelmodc_taper.scr` *in the demo directory reproduces the pictures.*

Using the parameters and `ntaper=n` enables the tapered boundaries with a taper length of n points. Besides those parameters specific boundaries must be put 'on' for tapering by using `left=4 right=4 top=4 bottom=4` . All enabled boundaries are using the same taper length and it is not possible to use different taper lengths for different boundaries. The number of taper points should be chosen such 1-2 times the main wavelength in the modeled data. The program calculates the number of taper points to be five times the wavelength $5\lambda_{tap} = 5\frac{max(c_p)}{f_{max}}$;

$$taper[ix] = \exp{-(0.30 * \frac{ix}{ntaper})^2}, \tag{45}$$

where ix is an integer ranging from 0 to $ntaper - 1$ and 0.30 is the taper factor. This taper factor can be

changed with the parameter `tapfact=`. A larger taper factor will make the taper go steeper to 0.0. In Figure 7 different `tapfact=` choices are shown. Note that if the taper is chosen too steep (larger than 0.5) the wavelet will already start reflecting from the beginning of the tapered boundary.
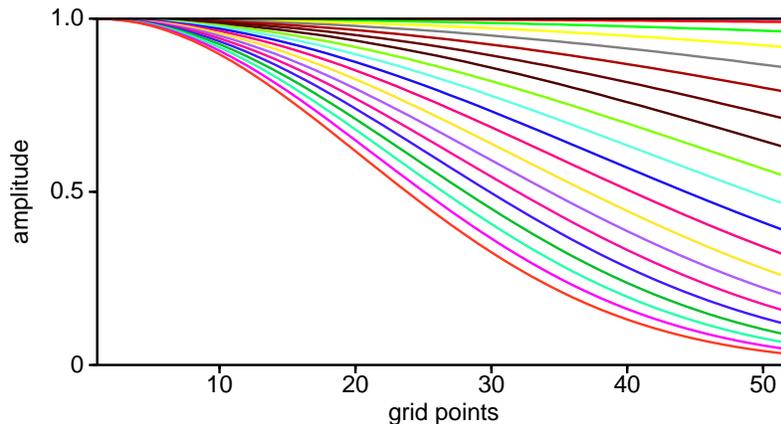
Figure 7: *The boundary taper as function of the* `tapfact=` *parameter is shown. The red line with the highest amplitude has* `tapfact=0.1`*, each line with a lower amplitude has a tap fact 0.1 larger (e.g. the green line has 0.2, the yellow line 0.3).*

Figure 6 shows the effects of the taper in a homogenous medium. It effectively suppresses the reflections from the sides of the model. The chosen taper length in this example is 200 points long. This is a large number of ;pints and used for illustration purposes. Using a taper is not a very efficient way of suppressing side reflections and there are plans to implement a better absorbing boundary method such as the Perfectly Matched Layer (PML) approach.

- Free surface (1)
  The free surface implementation for the acoustic scheme is straightforward and just sets the pressure field to zero on the free surface. For the elastic scheme the implementation is more involved and the free surface conditions (for the upper boundary) are :

  1. $\sigma_{zz} = 0$ is set
  2. $\sigma_{xz} = 0$ is implemented by defining an odd symmetry $\sigma_{xz}[iz] = -\sigma_{xz}[iz + 1]$.
  3. σ_{xx}: removed term with $\frac{\partial V_z}{\partial z}$, and add extra term with $\frac{\partial V_x}{\partial x}$, corresponding to free-surface condition for σ_{xx}. Other boundaries (left, right and bottom) are treated in a similar way.

The linearized equation of motion (Newton's second law) and equation of deformation (Hook's law) for the free surface become:

$$\sigma_{zz} = 0 = \frac{1}{\kappa}\frac{\partial V_z}{\partial z} + \lambda\frac{\partial V_x}{\partial x} \tag{46}$$

$$\sigma_{xz} = 0 = \mu\{\frac{\partial V_x}{\partial z} + \frac{\partial V_z}{\partial x}\} \tag{47}$$

In the FD code σ_{zz} is set to 0 at the free surface position $z = 0$. σ_{xz} is constructed in such a way that the difference around the free surface ends up to be zero:

$$\sigma_{xz}(0 - \tfrac{1}{2}\Delta z) = -\sigma_{xz}(0 + \tfrac{1}{2}\Delta z)$$
$$\sigma_{xz}(0 - 1\tfrac{1}{2}\Delta z) = -\sigma_{xz}(0 + 1\tfrac{1}{2}\Delta z)$$

Note that the location in the equations above are with respect to the grid for σ_{zz}. The trick for σ_{xz} is only needed for staggered grids to make the free surface of σ_{xz} on the same level as σ_{zz}.

For an expression of σ_{xx} on the free surface:

$$\sigma_{xx} = \frac{1}{\kappa}\frac{\partial V_x}{\partial x} + \lambda\frac{\partial V_z}{\partial z} \tag{48}$$

we substitute equation (46) $\frac{\partial V_z}{\partial z} = -\lambda\kappa\frac{\partial V_x}{\partial x}$ into and gives:

$$\sigma_{xx} = \frac{1}{\kappa}\frac{\partial V_x}{\partial x} - \lambda^2\kappa\frac{\partial V_x}{\partial x} \tag{49}$$

Using the parameters `left=1 right=1 top=1 bottom=1` enables a free surface boundary for all 4 sides.

```
if (bnd.top==1) { /* free surface at top */
    izp = bnd.surface[ixo];
    for (ix=ixo; ix<ixe; ix++) {
        iz = bnd.surface[ix-1];
        if ( izp==iz ) {
            /* clear normal pressure */
            tzz[ix*n1+iz] = 0.0;
        }
        izp=iz;
    }

    izp = bnd.surface[ixo];
    for (ix=ixo+1; ix<ixe+1; ix++) {
        iz = bnd.surface[ix-1];
        if ( izp==iz ) {
            /* assure that txz=0 on boundary by filling virtual boundary */
            txz[ix*n1+iz]   = -txz[ix*n1+iz+1];
            /* extra line of txz has to be copied */
            txz[ix*n1+iz-1] = -txz[ix*n1+iz+2];
        }
        izp=iz;
    }

    /* calculate txx on top stress-free boundary */
    izp = bnd.surface[ixo];
    for (ix=ixo; ix<ixe; ix++) {
        iz = bnd.surface[ix-1];
        if ( izp==iz ) {
            dp = l2m[ix*n1+iz]-lam[ix*n1+iz]*lam[ix*n1+iz]/l2m[ix*n1+iz];
            dvx = c1*(vx[(ix+1)*n1+iz]  - vx[(ix)*n1+iz]) +
                  c2*(vx[(ix+2)*n1+iz]  - vx[(ix-1)*n1+iz]);
            txx[ix*n1+iz] = -dvx*dp;
        }
        izp=iz;
    }
}
```

Placing a pressure source exactly on the free surface will not eject any energy into the medium and the resulting wavefield will contain only zero's. To overcome that you can use a F_z source (`src_type=7`) or place the pressure source one grid-point below the free-surface.

Note that you will always get a reflection from the free-surface. To summarise the effects:

– a P-source on a free surface can not put energy into the medium, and gives gathers with all zeros

– a F_z source (`src_type=7`) on the free-surface can put energy into the medium and is a good alternative for a P-source

– placing a P-source one-grid point below the surface, will simulate a dipole source. The $-$ part of the dipole coming from the reflection from the free surface.

– placing receivers, just one grid-point below the free surface, also gives a dipole receiver response.

– V_z receivers on a free suface measure an wavefield, P-receivers will not measure anything on a free surface.

To correct for the ghost of the source it is possible to de-ghost the measured response. This can be done with the program `basop option=ghost`. There is also a lot of literature about "source de-ghosting" (google search). The basop implementation is the most simple one.

- Rigid surface (3)
  The rigid boundary condition sets the velocities on the boundaries to zero. For the top surface these conditions are met by setting:

  – $V_x[iz] = 0.0$
  – $V_z[iz] = -V_z[iz + 1]$

  Setting the boundary parameters `left= right= top= bottom=` to 3 enables a rigid surface boundary for the selected boundary.

- PML (2)
  Only implemented for acoustic.

- Topography
  On the top of the model an irregular topography can be used. The density and velocity model must have zero-values above the defined topography. To place a source or receiver on the topography it is sufficient to place it at the correct lateral position above the topography. In the code the depth is searched for the first non-zero medium parameter and at that depth the source or receiver is placed. The parameter `sinkdepth=n` places the receiver position `n` grid-points below the found depth point on the topography. For the source position the parameter `sinkdepth_src=n` places the source position `n` grid-points below the found depth point on the topography. When the parameter `sinkvel=1` is used the receiver (not the source position) can also sink through a layer with a non-zero velocity. The velocity of the first receiver is used as the velocity to sink through to the next layer.

  For the elastic scheme the topography is implemented as described in Robertsson (1996) and Pérez-Ruiz et al. (2005). In those schemes the points at the topography layer are treated differently (depending on which side of the topography the free-surface is), such that the free surface is taken into account in the best possible way.

6.3.1 Recursive Integration PML: acoustic

The linearized equation of motion (Newton's second law) and equation of deformation (Hook's law) are given by:

$$\frac{\partial V_x}{\partial t} = -\frac{1}{\rho}\frac{\partial P}{\partial x}, \tag{50}$$

$$\frac{\partial V_z}{\partial t} = -\frac{1}{\rho}\frac{\partial P}{\partial z}, \tag{51}$$

$$\frac{\partial P}{\partial t} = -\frac{1}{\kappa}\{\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\}. \tag{52}$$

On PML introduces stretched-coordinate space (in the frequency domain) following Drossaert and Giannopoulos (2007) and Chew and Liu (1996):

$$\epsilon_x = 1 + \frac{\sigma_x}{j\omega} \tag{53}$$

$$\epsilon_z = 1 + \frac{\sigma_z}{j\omega} \tag{54}$$

$$\frac{\partial V_x}{\partial t} = -\frac{1}{\rho}\frac{1}{\epsilon_x}\frac{\partial P}{\partial x}, \tag{55}$$

$$\frac{\partial V_z}{\partial t} = -\frac{1}{\rho}\frac{1}{\epsilon_z}\frac{\partial P}{\partial z}, \tag{56}$$

$$\frac{\partial P}{\partial t} = -\frac{1}{\kappa}\{\frac{1}{\epsilon_x}\frac{\partial V_x}{\partial x} + \frac{1}{\epsilon_z}\frac{\partial V_z}{\partial z}\}, \tag{57}$$

The manner with which coordinate stretching causes losses in a medium can be easily understood as follows: Given a plane wave propagating in a lossless medium described by $\exp(jkx)$, a coordinate stretching is a change $x = x'(a + j\alpha)$. Then $\exp(jkx)$ becomes $\exp(jkax' - k\alpha x') = \exp(jkax') \exp(-k\alpha x')$. In other words, a wave becomes attenuative in the stretched x' coordinate if the stretching variable is complex (Chew and Liu, 1996). Introducing auxiliary variables

$$S_x = \frac{1}{\epsilon_x} \frac{\partial P}{\partial x} \rightarrow \qquad \frac{\partial V_x}{\partial t} = -\frac{1}{\rho} S_x, \tag{58}$$

$$S_z = \frac{1}{\epsilon_z} \frac{\partial P}{\partial z} \rightarrow \qquad \frac{\partial V_z}{\partial t} = -\frac{1}{\rho} S_z, \tag{59}$$

$$E_x = \frac{1}{\epsilon_x} \frac{\partial V_x}{\partial x} \quad E_z = \frac{1}{\epsilon_z} \frac{\partial V_z}{\partial z} \rightarrow \qquad \frac{\partial P}{\partial t} = -\frac{1}{\kappa} \{E_x + E_z\} \tag{60}$$

Substituting the stretch variables ϵ_x and ϵ_z into these auxiliary variables gives:

$$S_x + \frac{\sigma_x}{j\omega} S_x = \frac{\partial P}{\partial x}, \tag{61}$$

$$S_z + \frac{\sigma_z}{j\omega} S_z = \frac{\partial P}{\partial z}, \tag{62}$$

$$E_x + \frac{\sigma_x}{j\omega} E_x = \frac{\partial V_x}{\partial x}, \tag{63}$$

$$E_z + \frac{\sigma_z}{j\omega} E_z = \frac{\partial V_z}{\partial z} \tag{64}$$

To compute S_x, the other components can be done completely analogous, the following steps are carried out. The integration over time of S_x is approximated by using trapezoidal integration.

$$\frac{\sigma_x}{j\omega} S_x = \frac{\partial P}{\partial x} - S_x \tag{65}$$

$$\int_0^t \sigma_x S_x dt = \frac{\partial P}{\partial x} - S_x \tag{66}$$

$$\int_0^{n\Delta t} \sigma_x S_x dt \approx \frac{1}{2} \Delta t \sigma_x S_x^0 + \Delta t \sum_{t=1}^{n-1} \sigma_x S_x^t + \frac{1}{2} \Delta t \sigma_x S_x^n \tag{67}$$

$$\tag{68}$$

assuming $S_x^0 = 0$ results at time step n

$$S_x^n (1 + \frac{1}{2} \Delta t \sigma_x) + \Delta t \sum_{t=1}^{n-1} \sigma_x S_x^t = \frac{\partial P^n}{\partial x} \tag{69}$$

$$S_x^n = RA(\frac{\partial P^n}{\partial x} - \Delta t \Omega_x^{n-1}) \tag{70}$$

$$RA = \frac{1}{1 + \frac{1}{2} \Delta t \sigma_x} \tag{71}$$

$$\Omega_x^n = \Omega_x^{n-1} + \sigma_x S_x^n \tag{72}$$

The first few time steps in the PML regions are then computed as follows:

$$n = 0 : S_x^0 = RA(\frac{\partial P^0}{\partial x}) \tag{73}$$

$$\Omega_x^0 = \sigma_x S_x^0 \tag{74}$$

$$n = 1 : S_x^1 = RA(\frac{\partial P^1}{\partial x} - \Delta t \Omega_x^0) \tag{75}$$

$$\Omega_x^1 = \Omega_x^0 + \sigma_x S_x^1 \tag{76}$$

$$n = 2 : S_x^2 = RA(\frac{\partial P^2}{\partial x} - \Delta t \Omega_x^1) \tag{77}$$

$$\Omega_x^2 = \Omega_x^1 + \sigma_x S_x^2 \tag{78}$$

With these steps the auxiliary variables can now be computed and substituted into equations 58 for the PML regions.

6.3.2 Complex frequency shifted RIPML: acoustic

(This PML is not yet implemented.) Stretched-coordinate space (in the frequency domain) following Drossaert and Giannopoulos (2007):

$$\epsilon_x = \kappa_x + \frac{\sigma_x}{\alpha_x + j\omega} \tag{79}$$

$$\epsilon_z = \kappa_z + \frac{\sigma_z}{\alpha_z + j\omega} \tag{80}$$

$$\tag{81}$$

Following the same steps as before: Substituting these stretch variables ϵ_x and ϵ_z into the auxiliary variables 58 gives:

$$(\kappa_x \alpha_x + \kappa_x j\omega + \sigma_x)S_x = (\alpha_x + j\omega)\frac{\partial P}{\partial x}, \tag{82}$$

$$(\kappa_z \alpha_z + \kappa_z j\omega + \sigma_z)S_z = (\alpha_z + j\omega)\frac{\partial P}{\partial z}, \tag{83}$$

$$(\kappa_x \alpha_x + \kappa_x j\omega + \sigma_x)E_x = (\alpha_x + j\omega)\frac{\partial V_x}{\partial x}, \tag{84}$$

$$(\kappa_z \alpha_z + \kappa_z j\omega + \sigma_z)E_z = (\alpha_z + j\omega)\frac{\partial V_z}{\partial z} \tag{85}$$

To compute S_x, the other components can be done completely analogous, the following steps are carried out: divide by $j\omega$ results in

$$(\kappa_x \alpha_x + \kappa_x j\omega + \sigma_x)S_x = (\alpha_x + j\omega)\frac{\partial P}{\partial x}, \tag{86}$$

$$\frac{\kappa_x \alpha_x + \sigma_x}{j\omega}S_x + \kappa_x S_x = \frac{\alpha_x}{j\omega}\frac{\partial P}{\partial x} + \frac{\partial P}{\partial x}, \tag{87}$$

$$\int_0^t (\kappa_x \alpha_x + \sigma_x)S_x dt + \kappa_x S_x = \int_0^t \alpha_x \frac{\partial P}{\partial x} dt + \frac{\partial P}{\partial x} \tag{88}$$

$$\tag{89}$$

The integration over time of S_x and $\frac{\partial P}{\partial x}$ is approximated by using trapezoidal integration.

$$\int_0^{n\Delta t} aS_x dt \approx \frac{1}{2}\Delta t a S_x^0 + \Delta t \sum_{t=1}^{n-1} aS_x^t + \frac{1}{2}\Delta t a S_x^n \tag{90}$$

assuming $S_x^0 = 0$ and $\frac{\partial P^0}{\partial x} = 0$ results at time step n in

$$\Delta t \sum_{t=1}^{n-1}(\kappa_x\alpha_x + \sigma_x)S_x^t + \frac{1}{2}\Delta t(\kappa_x\alpha_x + \sigma_x)S_x^n + \kappa_x S_x^n = \Delta t \sum_{t=1}^{n-1}\alpha_x\frac{\partial P^t}{\partial x} + \frac{1}{2}\Delta t\alpha_x\frac{\partial P^n}{\partial x} + \frac{\partial P^n}{\partial x} \qquad (91)$$

$$\Delta t \sum_{t=1}^{n-1}(\kappa_x\alpha_x + \sigma_x)S_x^t + (\frac{1}{2}\Delta t(\kappa_x\alpha_x + \sigma_x) + \kappa_x)S_x^n = \Delta t \sum_{t=1}^{n-1}\alpha_x\frac{\partial P^t}{\partial x} + (\frac{1}{2}\Delta t\alpha_x + 1)\frac{\partial P^n}{\partial x} \qquad (92)$$

$$S_x^n = RA\frac{\partial P^n}{\partial x} - RB\Omega_x^{n-1} \qquad (93)$$

$$RA = \frac{(\frac{1}{2}\Delta t\alpha_x + 1)}{\frac{1}{2}\Delta t(\kappa_x\alpha_x + \sigma_x) + \kappa_x} \qquad (94)$$

$$RB = \frac{\Delta t}{\frac{1}{2}\Delta t(\kappa_x\alpha_x + \sigma_x) + \kappa_x} \qquad (95)$$

$$\Omega_x^n = \Omega_x^{n-1} + (\kappa_x\alpha_x + \sigma_x)S_x^n - \alpha_x\frac{\partial P^n}{\partial x} \qquad (96)$$

The first few time steps in the complex frequency shifted PML regions are then computed as follows:

$$n = 0 : S_x^0 = RA\frac{\partial P^0}{\partial x} \qquad (97)$$

$$\Omega_x^0 = (\kappa_x\alpha_x + \sigma_x)S_x^0 - \alpha_x\frac{\partial P^0}{\partial x} \qquad (98)$$

$$n = 1 : S_x^1 = RA\frac{\partial P^1}{\partial x} - RB\Omega_x^0 \qquad (99)$$

$$\Omega_x^1 = \Omega_x^0 + (\kappa_x\alpha_x + \sigma_x)S_x^1 - \alpha_x\frac{\partial P^1}{\partial x} \qquad (100)$$

$$n = 2 : S_x^2 = RA\frac{\partial P^2}{\partial x} - RB\Omega_x^1 \qquad (101)$$

$$\Omega_x^2 = \Omega_x^1 + (\kappa_x\alpha_x + \sigma_x)S_x^2 - \alpha_x\frac{\partial P^2}{\partial x} \qquad (102)$$

6.4   Source signature parameters

The parameter `file_src=` is used to define the filename of a SU file containing the source signature. This source signature should not have a large value at time $t = 0$, since this will represent a spike a $t = 0$, introduce high frequencies, and can make the modelling dispersive. If `file_src=` is not given the source sampling can be defined by setting `dt=`. To avoid numerical dispersion the maximum frequency content of the source wavelet must be limited (see section 1.2). The program tries to estimate the maximum amplitude in the frequency domain of the source signal, and based on this maximum determines if the modelling will be stable. The maximum frequency can also be given by the user with `fmax=` and will overrule the estimated maximum frequency. This overruling of the maximum frequency can be useful when the amplitude of the source signal is complex and the program can not make a good estimate of the maximum frequency. The parameter `fmax=` can also be used to overrule the programs error message (which stops the program) for dispersive modeling. By setting `fmax=` lower than the actual maximum frequency in the wavelet (estimated by the program) dispersion can be introduced. In some cases dispersion can be allowed, or dispersion is not relevant for the purpose of the modelling.

For example; a wavelet is created by :

```
makewave w=g2 fmax=45 t0=0.10 dt=0.001 nt=4096 db=-40 file_out=G2.su
verbose=1
```

This wavelet has a spectrum and time signal shown in Figure 8a and the maximum frequency estimated by the program is 48.8 Hz. To calculate the maximum frequency the program searches the peak frequency in the amplitude spectrum of the wavelet and then looks for the first frequency, larger than the peak frequency, where the amplitude is smaller than 0.0025 times the peak amplitude.

Note that the peak in the time domain of the wavelet in Figure 8a is time shifted with 0.1 seconds. Starting from $t = 0$ the amplitude of the wavelet is increased smoothly to its peak value and in this way avoids high frequencies.
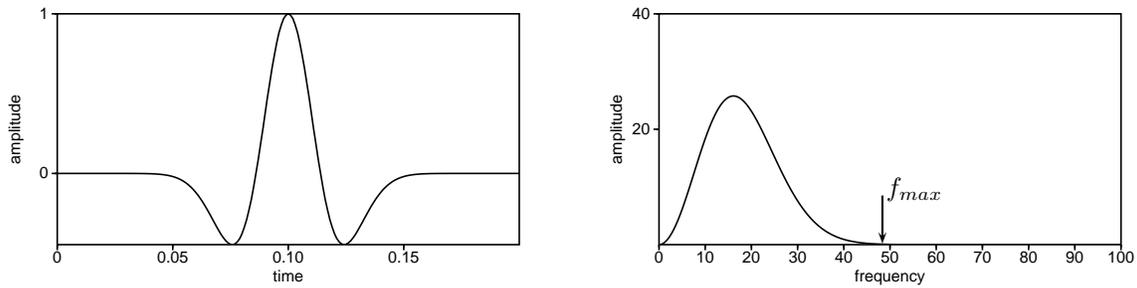
a) Ricker wavelet (left) and its amplitude spectrum (right).
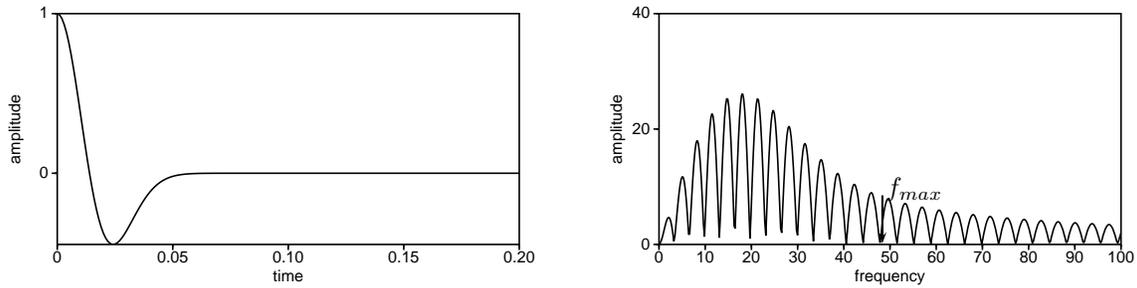
b) Truncated Ricker wavelet (left) and its amplitude spectrum (right).

Figure 8: *Wavelet and its amplitude spectrum. The maximum frequency in the wavelet is found by searching from the maximum amplitude to the first frequency amplitude $\leq 0.0025 * A_{max}$(indicated with an arrow). Note that due to the truncation at $t = 0$ in b) high frequencies are introduced and can cause dispersion. The script* `fdelmodc_rand.scr` *in the demo directory calculates the data and* `eps_for_manual.scr` *reproduces the pictures.*

A spike at t=0 (or any other time), or a truncated wavelet as shown in Figure 8b introduces high frequencies and will cause dispersion in the solution. So it is always recommended to check the wavelet on spikes and truncation before the modelling is started.

If the modelling is finished, and one has for example modeled a shot record with reflections, and would like to pick travel times on the peak of the wavelet, the 0.1 s, time shift of the wavelet should be taken into account! Good practice after modelling is to shift the peak of the wavelet back to $t = 0$ by using the parameter `rec_delay=0.1`. After the modeling one can also use other program, like `basop choice=shift shift=-0.1`, to shift the peak back to $t = 0$. When `rec_delay=` is set to a positive non-zero value the modelling time recorded in the receiver array will be the total modeling time decreased by the time of the `rec_delay`.

Noise signals are created (`wav_random=1`) by setting random values to the amplitude and phase of the source signal up to the given maximum frequency (`fmax=`). This signal is transformed back to the time domain and truncated in time to the desired source duration. Figure 9 shows 20 random signals in the time domain with varying source duration (average duration of 2.5 s `tlength=2.5`). Without any tapering this truncation in the time domain will introduce high frequencies. To suppress these high frequencies, the beginning and the end of the source signal are smoothly extrapolated (using cubic splines) to an amplitude value of 0.0. The bottom pictures in Figure 10 show a noise signal and its amplitude spectrum. This signal was constructed with a maximum frequency of 30 Hz. The start and beginning of the noise signal are smoothly starting and ending at amplitude zero. The red circles and lines shows how this signal is constructed. Despite the smooth start and ending of the signal the spectrum of the noise signal does continue after 30 Hz, but the amplitude after 30 Hz is so low that it does not give rise to severe dispersion in the modelling. The calculated noise source signatures are written to an SU file if the parameter `verbose>3` (see also Table 1). Note that if `file_src` is defined then `wav_random=0` is default set off. However, if `src_random=1` is used `wav_random=1` is default set on. The length (duration) of the random signal is chosen to be a random number (between 0.0 and 1.0 multiplied) by `tlength`. By setting `length_random=0` all random signals will have the same length given by `tlength`.
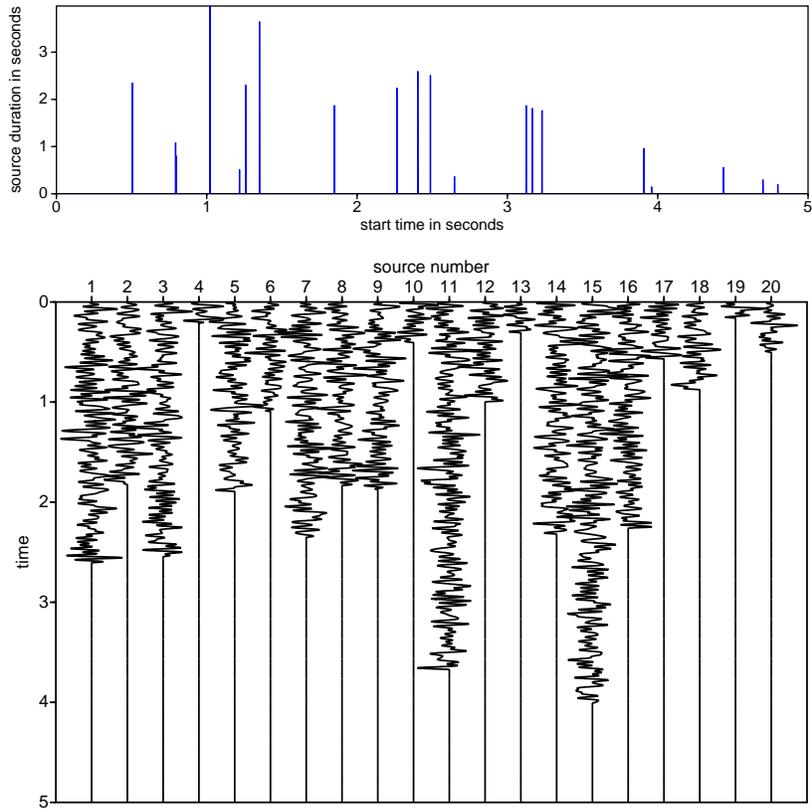
Figure 9: *Random source signatures with varying source duration (top picture). Note that the sources start at random times in the interval* `tsrc1=` *:* `tsrc2=`*. The script* `fdelmodc_rand.scr` *in the demo directory calculates the data and* `eps_for_manual.scr` *reproduces the pictures.*
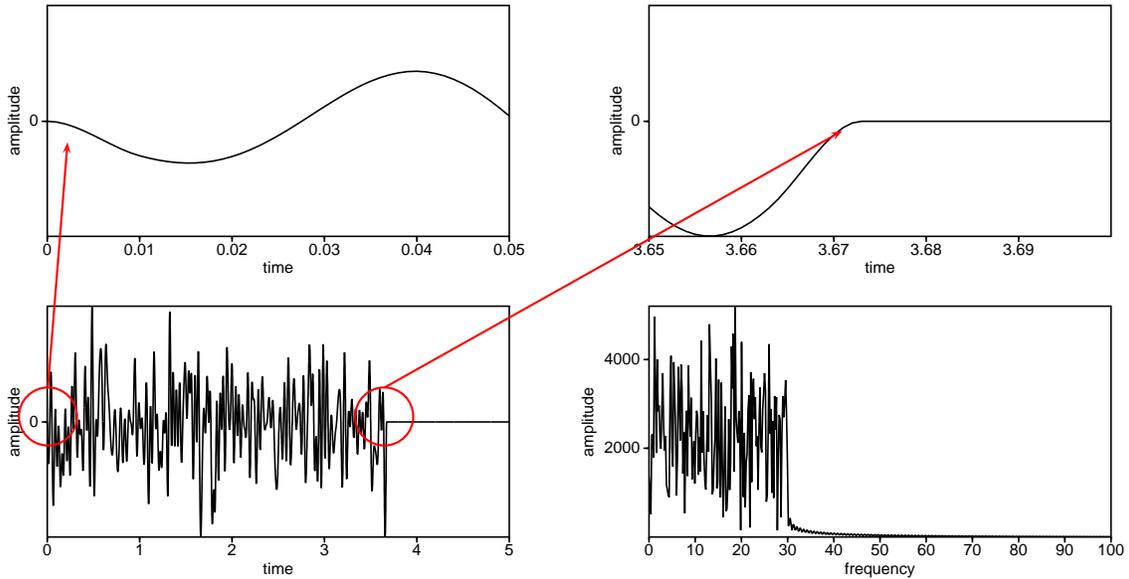
Figure 10: *Random source signature and its amplitude spectrum. The start and beginning of the source signature are smoothly (cubic spline) starting from and ending at amplitude 0. Despite this smooth transition the frequency spectrum of the signature still contains some energy after the defined maximum frequency. The script* `fdelmodc_rand.scr` *in the demo directory calculates the data and* `eps_for_manual.scr` *reproduces the pictures.*
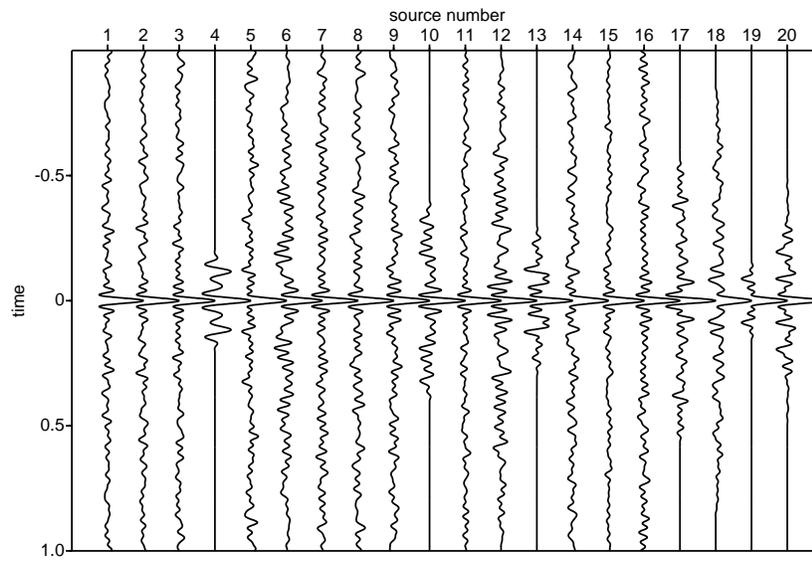
Figure 11: *Auto-correlated random source signatures with varying source duration normalized to the maximum amplitude per trace. The longest signals (source numbers 11 and 15) have the highest auto-correlation peak and best S/N ratio. The script* Figure17_19AppendixA.scr *in the FiguresPaper directory reproduces the pictures.*

The autocorrelation of the source signal gives an indication of the contribution of the individual sources to a Seismic Interferometry result. In Figure 11 the normalized auto-correlation of the signals in Figure 9 is shown. The longest signals will give a contribution, and has the highest signal (peak at $t = 0$) to noise ratio. The longer the source is active the more energy it will bring into the medium and the better the S/N ratio will be.

The cross-correlation of the source signals with each other gives how the different source signatures interfere with each other. Ideally the signatures should not interfere. Figure 12 show 100 source signatures and the cross-correlation. The diagonal is the auto correlation, and dominates the cross-correlation picture, indicating that the source are not correlated to each other.
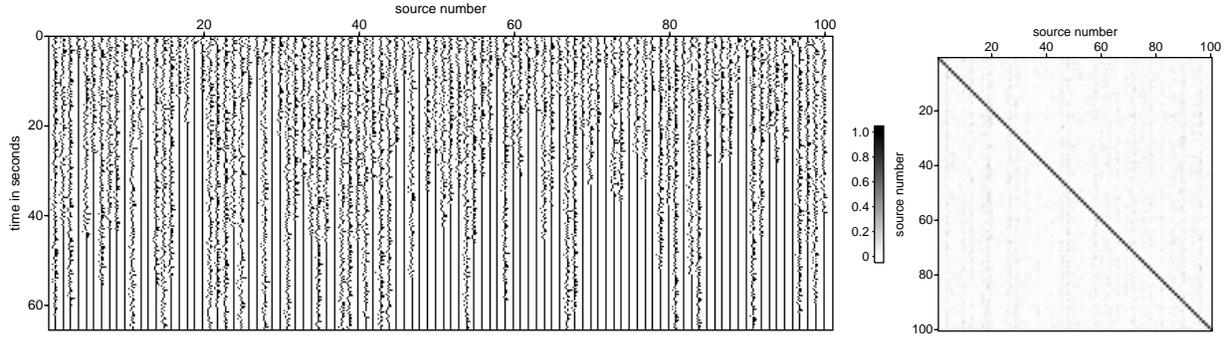
Figure 12: *The 100 source signatures on the left side have been cross-correlated with each other and the result is shown on the right side. The script* `cross.scr` *in the FiguresPaper directory reproduces these pictures.*

6.5 Source type and position parameters

The source amplitudes are added directly on the grid at the source position(s). The FD scheme solves the first order equations in(1) and the source amplitudes are added into these first order equations on the P, V_x or V_z grid. For example for a pressure source in an acoustic scheme (in a homogenous medium, to make the equations easier to read) the amplitudes are added at the P field only:

$$\frac{\partial P(x,z,t)}{\partial t} = -\frac{1}{\kappa}\{\frac{\partial V_x(x,z,t)}{\partial x} + \frac{\partial V_z(x,z,t)}{\partial z}\} + \frac{1}{\kappa}\delta(x-x',z-z')S(t), \tag{103}$$

$$\frac{\partial V_x(x,z,t)}{\partial t} = -\frac{1}{\rho}\frac{\partial P(x,z,t)}{\partial x}, \tag{104}$$

$$\frac{\partial V_z(x,z,t)}{\partial t} = -\frac{1}{\rho}\frac{\partial P(x,z,t)}{\partial z}, \tag{105}$$

where $S(t)$ represents the source signature injected at position x'. Substituting Hooke's Law in Newton's law gives the second order wave equation. Adding an amplitude on a grid point (representing a delta pulse $\delta(x-x',z-z')$) in the first order equations, results in a source term of $\partial_t\delta(x-x',z-z')$ in the right-hand side in the (second order) wave equation

$$\frac{\partial^2 P(x,z,t)}{\partial t^2}) - c_p^2\{\frac{\partial^2 P(x,z,t)}{\partial x^2} + \frac{\partial^2 P(x,z,t)}{\partial z^2}\} = \delta(x-x',z-z')\frac{\partial S(t)}{\partial t}. \tag{106}$$

To end up with a source injection in the wave equation, and not injection rates as in equation 106, the source signature is adjusted (in the frequency domain by $\frac{1}{-j\omega}$: integrating over time) before it is applied to the grid. The input parameter `src_injectionrate` can be set to choose between injection rate (set to 1) or injection (set to 0, which is the default).

For example when `src_injectionrate=0` a measured P response of a P source will measure the same source signature. The output file `src_nwav.su` (this file is written when `verbose=4`) contains the wavelet as it is being added to the grids in the FD code. Meaning that with `src_injectionrate=0` (default) it will be a time integrated version of the wavelet given by `file_src=`.

TODO. The general source function can be described as Wapenaar 1989 page 13:

$$S(t) = \frac{\partial \delta(t)}{\partial t} - \frac{\partial F_z}{\partial z}$$

(107)

Monopole source ($\frac{\partial \delta(t)}{\partial t}$)has same wavelet shape as Force source (dipole $\frac{\partial F_z}{\partial z}$)

6.5.1 Source type

The type of source and orientation is chosen with the parameters `src_type=` and `src_orient=` . The source type options are:

- 1=P for acoustic scheme placed on P grid and for elastic scheme on σ_{zz} and σ_{xx} grid with amplitude s

- 2=Txz for elastic scheme placed on σ_{xz} grid with amplitude s

- 3=Tzz for elastic scheme placed on σ_{zz} grid with amplitude s

- 4=Txx for elastic scheme placed on σ_{xx} grid with amplitude s

- 5=S-potential for elastic scheme placed on V_x and V_z grid with amplitude s (experimental)

- 6=Fx placed on V_x grid with amplitude $\frac{s\Delta x}{\rho}$

- 7=Fz placed on V_z grid with amplitude $\frac{s\Delta z}{\rho}$

- 8=P-potential for elastic scheme placed on V_x and V_z grid with amplitude s (experimental)

where s represents the amplitude of the source signal (from `file_src=` or a generated random signature). The source orientation can be changed from monopole to dipole, where the orientation of the dipole can also be changed. The implementation of the source orientation options are shown in Figure 13. The source orientation is only implemented for three types of sources: compressional (`src_type=1`), Txz (`src_type=2`), and pure shear source (`src_type=5`). For the other source types the `src_orient=` parameter has no effects and a monopole source is always used.
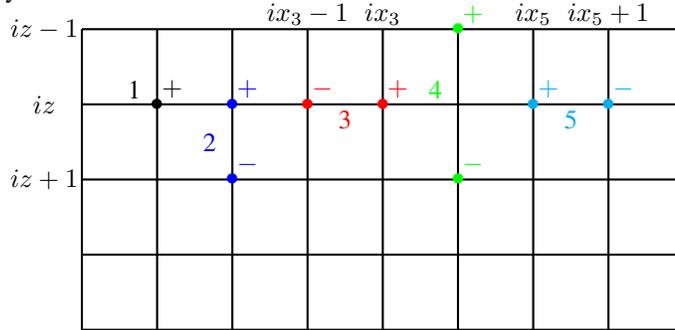
Figure 13: *Implementation of different source orientations on the grid. The different numbers (colors) show the options of parameter* `src_orient=`. *The* $+$ *and* $-$ *symbol represent the sign of the source amplitude added to the computational grid.*

Note, to check source-receiver reciprocity the dipole/monopole character of the source and receiver has to be taken into account as well. For example a dipole source with a V_z receiver (rvz) will be reciprocal with each other. The same for a monopole source and P receiver (rp). For a monopole source and V_z receiver this is reciprocal with a dipole source and and P receiver. For example run the code with: `src_orient=1` and select V_z receivers and after changing source and receiver x,z positions, set `src_orient=2` and use P receivers. Those measurements are source-receiver reciprocal with each other.

Pressure source

$$\frac{\partial P(x,z,t)}{\partial t} = -\frac{1}{\kappa}\{\frac{\partial V_x(x,z,t)}{\partial x} + \frac{\partial V_z(x,z,t)}{\partial z}\} + \delta(x-x',z-z')(\frac{1}{\kappa}S(t)),$$

(108)

$$\frac{\partial V_x(x,z,t)}{\partial t} = -\frac{1}{\rho}\frac{\partial P(x,z,t)}{\partial x},$$

(109)

$$\frac{\partial V_z(x,z,t)}{\partial t} = -\frac{1}{\rho}\frac{\partial P(x,z,t)}{\partial z},$$

(110)

Force source

$$\frac{\partial P(x,z,t)}{\partial t} = -\frac{1}{\kappa}\frac{\partial V_x(x,z,t)}{\partial x}, \tag{111}$$

$$\frac{\partial V_x(x,z,t)}{\partial t} = -\frac{1}{\rho}\frac{\partial P(x,z,t)}{\partial x} + \delta(x-x',z-z')(\frac{1}{\rho}S(t)), \tag{112}$$

$$\frac{\partial V_z(x,z,t)}{\partial t} = -\frac{1}{\rho}\frac{\partial P(x,z,t)}{\partial z}, \tag{113}$$

Potential S source (stype=5)

$$S_s(x,z,t) = \frac{\partial V_z(x,z,t)}{\partial x} - \frac{\partial V_x(x,z,t)}{\partial z}, \tag{114}$$

$$\tag{115}$$

```
vx[ix*n1+iz]     += src_ampl*sdx;
vx[ix*n1+iz-1]   -= src_ampl*sdx;
vz[ix*n1+iz]     -= src_ampl*sdx;
vz[(ix-1)*n1+iz] += src_ampl*sdx;
```

Potential P source (stype=8)

$$P_s(x,z,t) = \frac{\partial V_z(x,z,t)}{\partial z} + \frac{\partial V_x(x,z,t)}{\partial x}, \tag{116}$$

$$\tag{117}$$

```
vx[(ix+1)*n1+iz] += src_ampl*sdx;
vx[ix*n1+iz]     -= src_ampl*sdx;
vz[ix*n1+iz+1]   += src_ampl*sdx;
vz[ix*n1+iz]     -= src_ampl*sdx;
```

To have correct amplitudes between the different source types and independence of the discretization in the finite difference code $(\Delta t, \Delta x)$, scaling factors are applied to the input source wavelet amplitudes.

src-type	amplitude
1,2,3,4	$\frac{\Delta t}{\Delta x^2}2.0C_p^2\rho$
5,6,7,8	$\frac{\Delta t}{\Delta x^2}\frac{2.0}{\rho}$

The factor 2 is added to be compliant with the defined Greens functions.

TODO: the relative amplitudes of the sources in the elastic scheme are correct, but still have to check the (absolute) source amplitude factors in the elastic scheme (compare them with analytical Green's functions in homogenous medium).

The parameter `grid_dir` reverses the time of the source wavelet and can be used to back-propagate a recorded source field into the medium.

6.5.2 Source positions

Sources can be defined in three different ways in the program. A source can be placed on single grid point, can be placed on many grid points which are starting at the same time creating an areal source field (or plane wave), or sources can be placed at many random positions on the grid starting at random positions in time.

Single or regular shot distribution The position of a single source is set by the parameters `xsrc=` `zsrc=`. The parameters `nshot=` together with `dxshot=` and `dzshot=` determine how many shots are modeled and the position for each shot. For every new shot of the `nshots` to model the xsrc and zsrc positions are adjusted with dxshot and dzshot. The loop over the number of shots to model is

```
for (is=0; is<shot.n; is++) {
      shot->x[is] = xsrc+is*dxshot;
      shot->z[is] = zsrc+is*dzshot;
}
```

For example `xrsc=50 zsrc=0 dxshot=15 dzshot=0 nshot=5` will successively model 5 shots at the positions (50,0) (65,0) (80,0) (95,0) and (110, 0). In Figure 14a the black dots represent the source positions which have been defined using a regular shot position. This means that for every black dot in Figure 14a a shot is calculated.
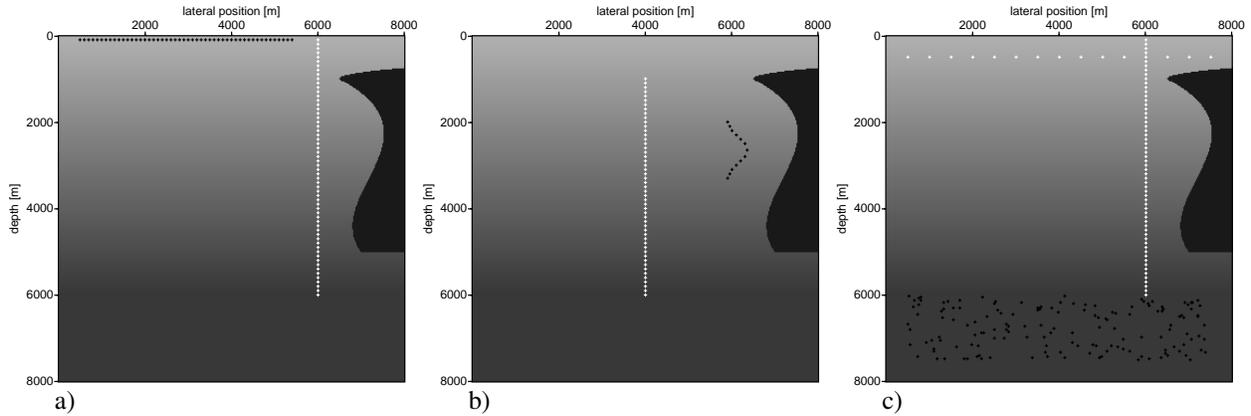
Figure 14: *Source (black dots) and receiver (white dots) positions for different choices of the positions parameters. In*

a)   `xrcv1=6000 xrcv2=6000 dxrcv=0 zrcv1=100 zrcv2=6000 dzrcv=100 xsrc=500 zsrc=100 nshot=50 dxshot=100 dzshot=0` *has been used, for*

b)   `xsrca=5900,5950,6000,6100,6200,6300,6350,6300,6200,6100,6000,5950,5900 zsrca=2000,2100,2200,2300,2400,2500,2650,2800,2900,3000,3100,3200,3300 xrcv1=4000 zrcv1=1000 xrcv2=4000 zrcv2=6000 dzrcv=100 dxrcv=0` *and for*

c)   `xrcv1=6000,500 xrcv2=6000,7500 dxrcv=0,500 zrcv1=100,500 zrcv2=6000,500 dzrcv=100,0 src_random=1 nsrc=150 xsrc1=500 xsrc2=7500 zsrc1=6000 zsrc2=7500` *. The script* `fdelmodc_srcrec.scr` *in the demo directory reproduces the pictures.*

Source arrays   An array of sources (or an areal source) is defined with the parameters `xsrca= zsrca=`. The parameters define the position of the sources. For example `xrsca=50,55,67,40,12 zsrca=0,10,7,8,10` defines an areal source consisting of 5 shot positions. The shots at these positions are all fired simultaneously. In Figure 14b the black dots represent the source positions, which have been defined using a source array. This means that all black dots in b) will be fired simultaneously an only one shot is calculated.

You can also use multiple sources simultaneously by placing multiple source wavelets (with different amplitude and frequency) into the `file_src` file and set the parameter `src_multiwav=1`. The position of these sources are read from the Seismic Unix header values; gx for the x-postion, and gelev for the z-position of the source. Setting parameter `src_at_rcv=0` will use the sx for the x-postion and selev for the z-position. Note that these setting of the scalco (gx) and scalel (gelev) factor in the Seismic Unix file. To fire multiple sources (more than 2) after each other, and construct one receiver gather that contains the wavefields generated by these sources, can be achieved by adding a time delay to the traces of `file_src` and set `src_multiwav=1`. All traces will start at the same time in the modeling program. In this case some trace will only have a non-zero amplitude in the beginning and become active (non-zero) after a certain timing.

Plane wave   A plane wave can be defined by using the parameters: `plane_wave= nsrc= src_angle= src_velo= nsrc= src_window=`. The plane wave is implemented by placing a horizontal array of `nsrc=` sources placed on every grid position symmetric around `xsrc=` at the horizontal depth given by `zsrc=`. The angle (ray-parameter) of the plane wave is defined by `src_angle= src_velo=` and is implemented by adding time delays to the shot positions on the horizontal array. Figure 17b shows the plane wave source positions as blue dots in the model.

Source random positions   `src_random= nsrc= xsrc1= xsrc2= zsrc1= zsrc2= tsrc1= tsrc2= tactive= tlength= amplitude= distribution= seed=`

33

When the parameter `src_random=1` is used, `nsrc=` random source positions will be created with positions between `xsrc1=` : `xsrc2=` and `zsrc1=` : `zsrc2=`. During the model time `tmod=`, sources will start and at random times in the interval `tsrc1=` : `tsrc2=` and contribute to the calculated wavefield. The maximum signal length is defined by `tlength=` and the resulting average length = `tlength`/2. Note that if `tlength=` is larger than `tsrc2-tsrc1` then for some sources the time the source is being active (`tlength`) will be truncated to `MIN(tsrc1+tlength, tactive)`. The parameter `tactive=` gives the maximum modeling time the sources are being active. After `tactive=` no sources are being active anymore. The source time settings allows modeling of many random source positions being active by running just one modeling. The amplitude distribution of the sources is default set to 0, meaning that all sources have the same amplitude. Defining for example `amplitude=10` will introduce an amplitude distribution for the different source with a maximum amplitude of 10. This distribution can be made flat `distribution=0` or Gaussian `distribution=1`. The `seed=` parameter can be used to generate different random sequences, if the same `seed` value is used modeling results can be reproduced. In Figure 14c random source positions are used for the lower part in the model and visible as small black dots in the Figure.

6.6   Receiver, Snapshot and Beam parameters

6.6.1   Receiver, Snapshot and Beam type

The type of wavefield component of the receiver, snapshot and beam recordings which will be written to file can be selected by using `sna_type_*=` for snapshots and beam fields and `rec_type_*=` for the receivers, where * is one of the following characters:

- p: P registration with file extension `_rp` only for acoustic scheme

- vz: V_z registration with file extension `_rvz`

- vx: V_x registration with file extension `_rvx`

- txx: T_{xx} registration with file extension `_rtxx`

- tzz: T_{zz} registration with file extension `_rtzz`

- txz: T_{xz} registration with file extension `_rtxz`

- pp: P potential registration with file extension `_rP`

- ss: S potential registration with file extension `_rS`

- ud: up- and down-going decomposition with file extension `_ru`, `_rd` (only for receiver arrays)

In the acoustic scheme only `rec_type_p=`, `rec_type_vz=` and `rec_type_vx=` can be used. In elastic media there is no pressure component, so you can not record it. However, if you set-up an elastic model and put a water layer on top of that elastic model you can place a receiver in that water layer and the Tzz component of the elastic modeling scheme will contain the pressure field in the water. We have made a comparison between the T_{zz} component of the elastic field in a water layer with the P field in an acoustic layer and they are identical.

The potential wavefields P and S for the elastic scheme are based on divergence and curl (rotation), respectively, given by:

```
rec_pp[irec*rec.nt+isam] = (vx[ix2*n1+iz]-vx[ix*n1+iz] +
                            vz[ix*n1+iz2]-vz[ix*n1+iz])/mod.dx;


rec_ss[irec*rec.nt+isam] = (vx[ix2*n1+iz2]-vx[ix2*n1+iz] -
                            (vz[ix2*n1+iz2]-vz[ix*n1+iz2]))/mod.dx;
```

The up- and down-going wave fields make use of acoustic decompositions operators Wapenaar (1998). The P and V_z fields needed to do this decomposition are automatically enabled when `rec_type_ud=1` is chosen. In the program all grid points in x, for the chosen receiver-depth level, are stored into memory for the P and V_z fields. The decomposition is carried out, in the wavenumber-frequency domain, on these finely sampled fields. To avoid artefacts from the edges of the model an angle filter is applied in the wavenumber-frequency domain. The cut-off angle in this filter is estimated from the data at the receiver level. The maximum angle present in the receiver

data is estimated by summing along $k_x = k\sin(\alpha)$ lines in the wavenumber-frequency amplitude spectrum. The maximum angle in the data is then selected as the angle at the average amplitude (this is just an simple method and can be improved). Figure 15 shows the energy in the angles in the wavenumber domain for the P-field (based on the decomposition.scr in the demo directory). In this example the average energy is 200 and leads to a maximum angle of 67°. The calculated angle file is written to output file `anglerp.su` when the `verbose` option is set to 4 or larger. The use of the calculated angle can be overruled by setting the parameter `kangle=`.

a) V_z-field   b) P-field   c) up-going field   d) down-going field

e) angle-energy plot.
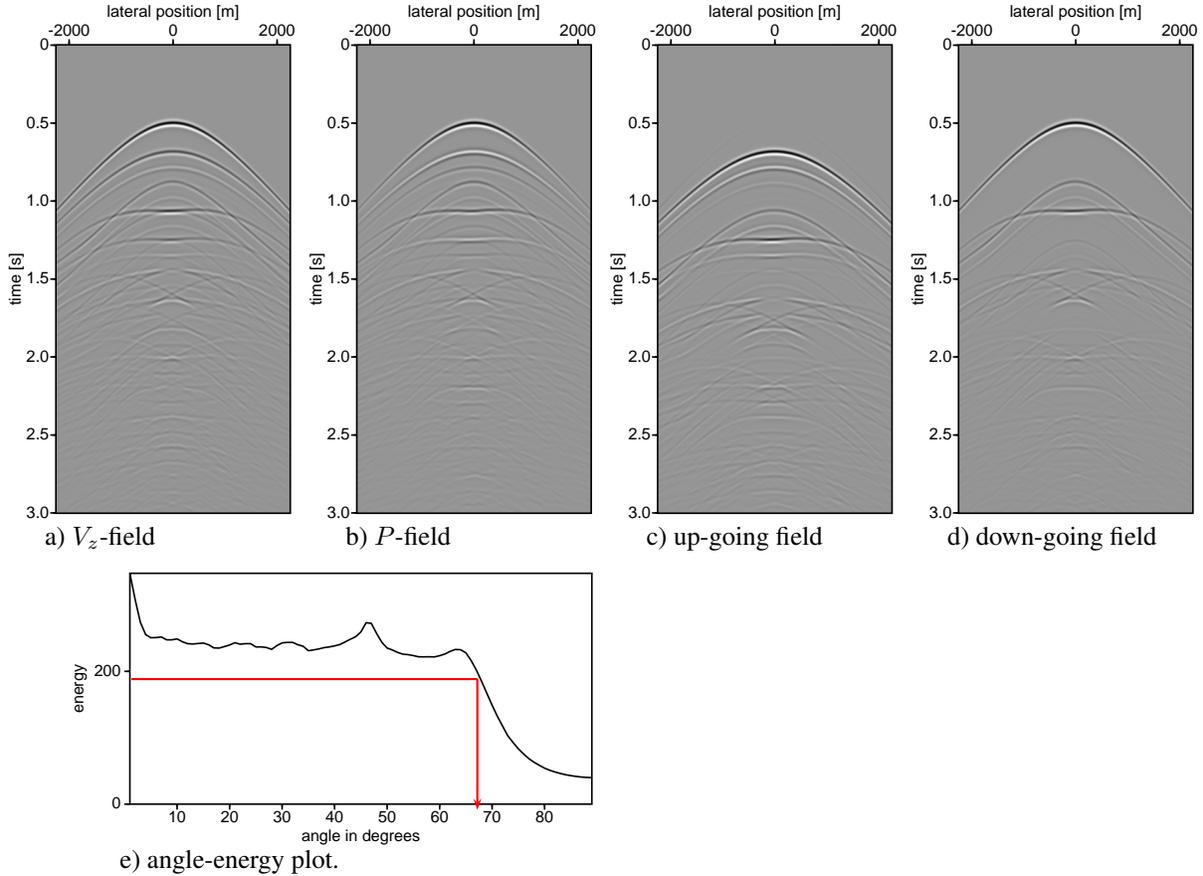
Figure 15: *The P (a) and V_z (b) fields and the decomposed up (c) and downgoing (d) fields from example demo/decompostion.scr. Selection of maximum angle (e) to filter the decomposition operator in $k_x - \omega$ domain.*

The receiver time sampling rate and number of time samples are defined by `dtrcv=` and `rec_ntsam=`. Usually `dtrcv=` is chosen larger than the `dt` of the modeling. Note that `dtrcv=` misty be chosen as an integer multiple of the timne step of the modeling `dt`. The recording can start later than the modeling, for example to compensate for a time-delay in the source wavelet (see section 6.4) by using `rec_delay=`.

Seismic unix (and SEGY) uses an unsigned shot integer (16 bytes) in the header section of a trace to store the number of samples in a trace. The maximum value of this short integer is 65536, so more samples can not be stored in a single trace. In passive seismic modeling the recording times can be very long and one easily exceeds 65536 samples in a trace. For long recordings multiple files are created each with `rec_ntsam=` samples per trace in a file. The names of the output files are numbered starting with `001` and this number is placed in the filename just before the name of the receiver type (for example `shotA_001_rp.su`, `shotA_002_rp.su`, `shotA_003_rp.su`, `...`).

6.6.2 Receiver positions

Receiver linear array(s) As the name already tells, linear receiver arrays are receivers defined on lines. Using the parameters `xrcv1=`, `xrcv2=` `zrcv1=` and `zrcv2=` multiple lines can be defined. Using `xrcv1=100`, `xrcv2=500` defines one receiver line starting at x-position 100 until 500 m and uses the default z-position which is the top of the model. The distance between the receiver in the line is defined by `dxrcv=` or `dzrcv=`.

Defining multiple lines, for example on multiple depth levels, is accomplished by using `xrcv1=100,100` `xrcv2=500,500` `zrcv1=100,200`, `zrcv2=100,200` and `dxrcv=10,10` defines two receiver lines from 100 to 500 m in the lateral direction; one on a depth level of 100 m and another one at 200 m depth. Note that in defining multiple linear arrays the number of arguments in the parameters must be the same. The distance between the receivers can also be different for the different lines by using `dxrcv=10,5` or `dzrcv`. If only one value is used for the distance parameters (one from `dxrcv=` `dzrcv=`) then all lines will use this distance. A receiver line is defined by the end points: start (`xrcv1,zrcv1`) and end (`xrcv2,zrcv2`) and a distance between the receivers. This distance can be given by `dxrcv=` or `dzrcv`, if both are given then `dxrcv=` is used and `dzrcv` is calculated in the program. However, if `dxrcv=0` and `dzrcv` is also given(and not zero) then the given `dzrcv` is used. Multiple receiver arrays will be placed into the same receiver file in the same order you have defined them. Another example to place a receiver array at the surface and two vertical arrays at $x = 500$ and $x = 1500$ you can use the following parameters:
`xrcv1=-2500,500,1500` `xrcv2=2500,500,1500` `zrcv1=0,0,0` `zrcv2=0,1000,1000` `dxrcv=20,0,0` `dzrcv=0,10,10`

Receiver array   An array of receivers at specific points in the medium is defined with the parameters `xrcva=` `zrcva=`. The parameters define the position of the receivers. For example `xrcva=50,55,67,40,12` `zrcva=0,10,7,8,10` defines a receiver array of 5 positions.

Receivers on circle or ellipse   To put the receivers on a circle the following parameters can be used: `rrcv=` for the radiation of the circle, `oxrcv,ozrcv` sets the origin (in meter, not grid points) of the circle and `dphi` the distance between the receivers in angle. To make an ellipse use the additional parameter `arcv=`. This sets the size of the vertical arc-length and `rrcv=` represents then the horizontal arc-length. The distance between the receivers on the ellipse are (numerically) made equidistant (the `dphi` parameter is in the ellipse option only used to compute the number of receivers (=360/`dphi`). Note that due to the grid of the wavefields the receivers are not placed on exactly a circle/ellipse, but to the closest grid point. The option `rec_int_p` or `rec_int_vz` `rec_int_vx` set to 3 will interpolate (bi-linear) the fields to the exact position of the receivers on the circle. When `rrcv` is used the parameters `rec_int_p` and `rec_int_vx`, `rec_int_vz` are automatically set to 3.

Receivers from and ASCII text file   *implemented by Max Holicki*
Reading many receiver positions with `xrcva=` `zrcva=` can become cumbersome when many receivers are needed. An option has been build in to read receiver locations from a text file.
With the argument `rcv_txt=` one can specify an ASCII test file from which receiver coordinates should be loaded. The file must have the following format:

```
25 10
50 15
75 20
```

each line in this file contains a receiver location (x,z). Loading this file would place 3 receivers at (25,10), (50,15), and (75,20).
A demo script to demonstrate the functionality is called `RcvTextInput.scr`.

6.6.3   Interpolation of receiver positions

The parameters `rec_int_vx` and `rec_int_vz` have the options

```
rec_int_vx=0  ..... interpolation of Vx receivers",
                 – 0=Vx->Vx (no interpolation)",
                 – 1=Vx->Vz",
                 – 2=Vx->Txx/Tzz (P)",
                 – 3=Vx->receiver position
rec_int_vz=0 ...... interpolation of Vz receivers",
                 – 0=Vz->Vz (no interpolation)",
                 – 1=Vz->Vx",
                 – 2=Vz->Txx/Tzz (P)",
                 – 3=Vz->receiver position
```

for reading the wavefield at the receiver positions from the staggered grid positions.

- `rec_int_vz=1` interpolates V_z to the V_x position and makes use of the 4 surrounding V_z points (blue colored items in Figure 16)

- `rec_int_vz=2` interpolates V_z to the σ_p position and uses two 2 (top and down) V_z points (cyan colored items in Figure 16)
  interpolates σ_{xz} to the σ_p position and makes use of the 4 surrounding σ_{xz} points (black colored items in Figure 16)

- `rec_int_vx=1` interpolates V_x to the V_z position and makes use of the 4 surrounding V_x points (green colored items in Figure 16)

- `rec_int_vx=2` interpolates V_x to the σ_p position and uses two 2 (left and right) V_x points (magenta colored items in Figure 16)
  interpolates σ_{xz} to the σ_p position and makes use of the 4 surrounding σ_{xz} points (black colored items in Figure 16) Using `rec_int_vz=2` and `rec_int_vx=2` puts V_z, V_x and σ_{xz} to the σ_p position.

- `rec_int_vx=3` or `rec_int_vz=3` interpolates all fields to the exact receiver positions given by the user using bi-linear interpolation. In this case the receiver positions do not have to lie on a grid position (yellow colored items in Figure 16). This option is turned on by default when receivers on a circle are defined.
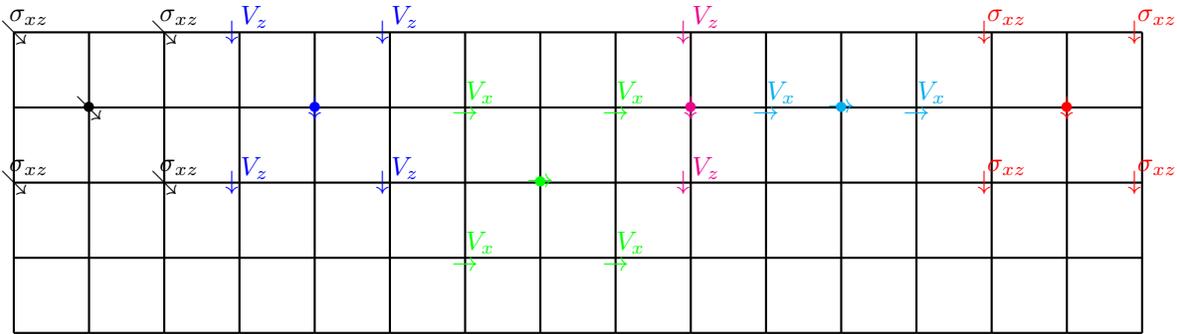
Figure 16: *Receiver interpolation options in the elastic staggered calculation grid. V_z, V_x represent the particle velocity of the wavefield in the z and x direction respectively and $\sigma_p(\sigma_{xx}$ or $\sigma_{zz})$, σ_{xz} represent the stress fields.*

6.6.4 Snapshots and Beams

During modeling time snapshots of complete wavefields can be written to a file `file_snap=snap.su` on disk and, for example, used in a movie showing wave propagation through a model. To define time snapshots the start-time of the first snapshot `tsnap1=`, the time of the last snapshot `tsnap2=`, and the time interval between the snapshots `dtsnap1=` have to be defined. Using the default parameters the output size of one snapshot is as large as the gridded model file. To reduce the space of the snapshot, since it might not needed to have the snapshot on the same fine grid as the modeling, `dxsnap=` and `dzsnap=` can be used to increase the grid distance in the x and z direction. If one is only interested in a certain area the parameters `xsnap1=`, `xsnap2=`, `zsnap1=`, and `zsnap2=` can be used to define that area.
The finite-difference scheme is also staggered in time and the parameter `sna_vxvztime` defines which registration of vx/vz times is used in the snapshots.

- 0=previous vx/vz relative to txx/tzz/txz at time t

- 1=next vx/vz relative to txx/tzz/txz at time t

Beams represent the energy of the wavefield during the complete model time. Beams are calculated as the square root of the quadratic field quantities and can be used to investigate how energy propagates through a model. During every time step the 'energy' is calculated and added to the beam array. Beams are enabled by setting `beam=1` and

the same parameters as for the snapshots are used to define grid distance `dxsnap=` and `dzsnap=`, area of interest, and type of wavefield component. Note that the beam calculation is done for every time-step and is an expensive computational operation, the total compute time can easily increase by 50% or more.

6.7 Verbose

The parameter `verbose` prints messages and produces additional files during the running of the program. Table 1 shows the kind of messages and the extra files printed using different values for `verbose`. Those messages and files contain extra information for the user. The output files produced by different setting of the verbose parameter are:

- `src_nwav.su`: file which contains the source signatures. For the noise sources each trace in this file contains the noise signal used in the modeling. Note that this file can become large for long wavelengths

- `SrcRecPositions.su`: SU file with the same size as the input model files. At every source position a square (of 5x5 grid points) with value 1 is positioned and at every receiver position a square with value -1 is placed. This file can be used to overlay (after scaling) the source and receiver positions on a velocity model. For example the following command:

  ```
  suop2 SrcRecPositions.su vel2_edge_cp.su w1=6000 w2=1.0 op=sum | suximage
  bclip=6000 wclip=0
  ```

  adds source and receiver positions to the P-wave velocity file, where the weight factor `w1` is set larger than the maximum velocity in the P-wave velocity file.

- `srcTimeLengthN=ns.bin`: 32 bits floating point binary file with ns samples where each sample value gives the start time of the source.

- `SrcPositionsNSRC.txt`: ASCII file which contains the source positions where NSRC is the number of sources.

- `RcvPositionsNRCV.txt`: ASCII file which contains the receiver positions where NRCV is the number of receivers.

- `src_ampl.su`: when the `amplitude` parameter is used this file contains the amplitude distribution of the sources.

setting	messages printed to stdout	files written to disk
0	no messages only warnings	no files
1	model, source, receiver, snapshot and stability info	no files
2	same as 1	no files
3	+ file-info from model and wavelet files	src_ampl.su
4	+ receiver grid positions, source positions	+ srcTimeLengthN=ns.bin, SrcRecPositions.su, src_nwav.su
>4	+ source grid points, amplitude and start time, topography surface points	same as 4

Table 1: *The files and messages produced by different values of the* `verbose` *parameter. The file src_ampl.su contains the amplitude variation of the (random) source signals when the parameter* `amplitude=` *is set to a non-zero value. srcTimeLengthN=ns.bin is a binary file with ns samples which give the start time of the sources. SrcRecPositions.su is a SU file with the same size as the input model files and at every source position a cross with value 1 is positioned. The other positions in the file are set to zero. File src_nwav.su contains the generated random source signals. Note that this file can become large for long wavelengths* (`tlength=`)
.

7 Examples to run the code

The demo directory contains scripts which demonstrate the different possibilities of the modeling program. In the subsections below most demo script are explained and results are shown.

To reproduce the Figures shown in the GEOPHYICS manuscript "Finite-difference modeling experiments for seismic interferometry" the scripts in FiguresPaper directory can be used. Please read the README in the FiguresPaper directory for more instructions and guidelines.

7.1 Example for plane waves: `fdelmodc_plane.scr`

Modeling plane waves can easily be done by setting the plane wave option `plane_wave=1`. The demo script `fdelmodc_plane.scr` is set-up for an elastic medium. The number of sources used to make the plane wave is controlled with `nsrc=`. The centre of the plane wave will be positioned at the source position defined by the parameters `xsrc= zsrc= `. Left and right of this central position `(nsrc-1)/2` sources are placed on every grid point in the lateral (x) direction. In the example below the plane wave has been given an angle of 5° using `src_velo=1800 src_angle=5`. The script file `demo/fdelmodc_plane.scr` contains all the commands used to make the plane waves based figures shown in this subsection. The runtime of the script is approximately 120 seconds.

The gridded model is shown in Figure 17 and contains one flat smooth interface, a vertical velocity gradient, and a curved interface.

```
../fdelmodc \
    file_cp=$filecp file_cs=$filecs file_den=$filero \
    ischeme=3 \
    file_src=wavelet.su verbose=1 \
    file_rcv=rec.su \
    file_snap=snap.su \
    xrcv1=0 xrcv2=2000 dxrcv=15 \
    zrcv1=400 \
    rec_type_vx=1 rec_type_pp=1 rec_type_ss=1 rec_int_vx=1 \
    dtrcv=0.004 \
    xsrc=1000 zsrc=1700 nshot=1 plane_wave=1 nsrc=301 \
    src_type=1 tmod=3.0 src_velo=1800 src_angle=5  \
    ntaper=120 \
    left=4 right=4 bottom=4 top=4 \
    tsnap1=0.1 tsnap2=3.0 dtsnap=0.1 \
    sna_type_ss=1 sna_type_pp=1 verbose=4
```

The receivers are placed at $z = 400$ m. The measured particle velocity field V_z and the calculated P- and S-wave potentials are shown in Figure 18a,b, and c respectively. Snapshots, where three modeling times are added together, are shown in Figure 18d,e, and f. In the S-potential snapshot (Figure 18f) it is observed that the S-waves are firstly occurring when the P-wave hits the curved reflector. Note also the better resolution of the S-potential. In the run script the verbose option is set to 4, which means that auxiliary files are also written to disk. These files contain information information about the used source and receiver positions and can be used to plot the source and receiver positions in the model.

a)Model

b)Model with source and receiver positions
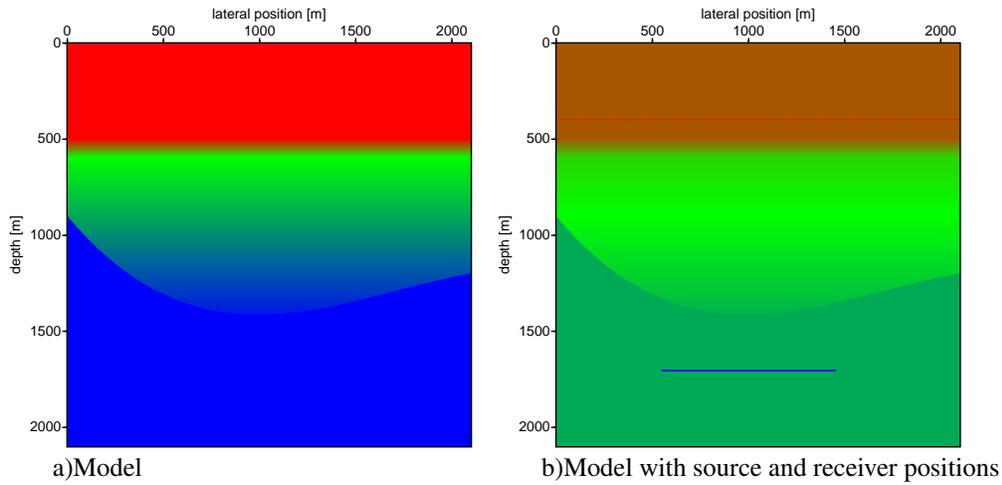
Figure 17: *Gridded model used to model a plane wave at $x = 1000, z = 1700$. The receivers are placed at depth level $z = 400$ m, just outside the taper area at the top, which ends at $z = 360$ m. Source (blue dots) and receiver (red dots) positions are shown in b).*

a) V_z receivers

b) P-potential receivers

c) S-potential receivers

d) V_z snapshots

e) P-potential snapshots

f) S-potential snapshots
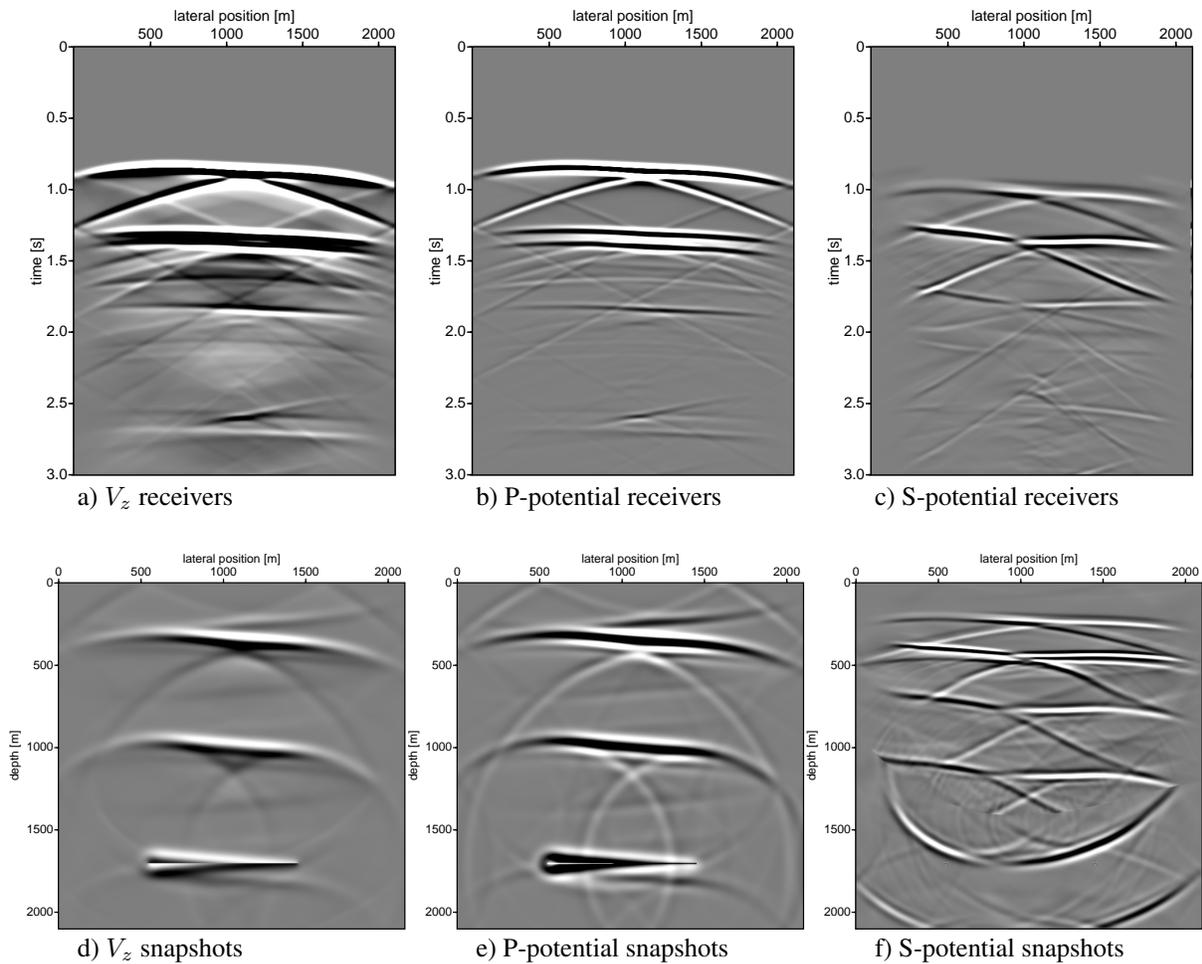
Figure 18: *Panel a,b, and c show three different types of receiver measurements at depth $z = 400$ m. Panel d, e, and f show three snapshots at $t = 0.5, 0.9, 1.3$ for different types of wavefield components.*

7.2 Example for viscoelastic media: `fdelmodc_visco.scr`

Visco-elastic modeling is enabled by choosing the parameter `ischeme=4`. In the demo script `fdelmodc_visco.scr` a model is created with `makemod` for the density and the P and S wave velocities. Besides the model which describes the medium parameters another gridded model is created with makemod for the Q-values (Q_p and Q_s) of the medium. The parameters of makemod are set-up to be used to create cp,cs and rho gridded model files. However, it can be (ab-)used for building any gridded file, where the values of cp, cs and rho just get another meaning. To create the Q-values with makemod the cp velocity is used as Q_p and the cs value is used as Q_s, the density value is not used. In this way the Q-model is created and used in `fdelmodc`. The results of running the demo script is shown in Figure 19.

Another trick has been used to subtract the direct field of the measured recordings. This is done by running fdelmodc a second time, and this time in a homogeneous medium with the velocity of the layer were the receivers are placed. Then the recorded fields of the direct field is subtracted from the completed field resulting in only the reflected field. Running the `fdelmodc_visco.scr` takes about 10 minutes.

a) V_x recordig	b) V_x recording	c) S-potential recording	d) S-potential recording
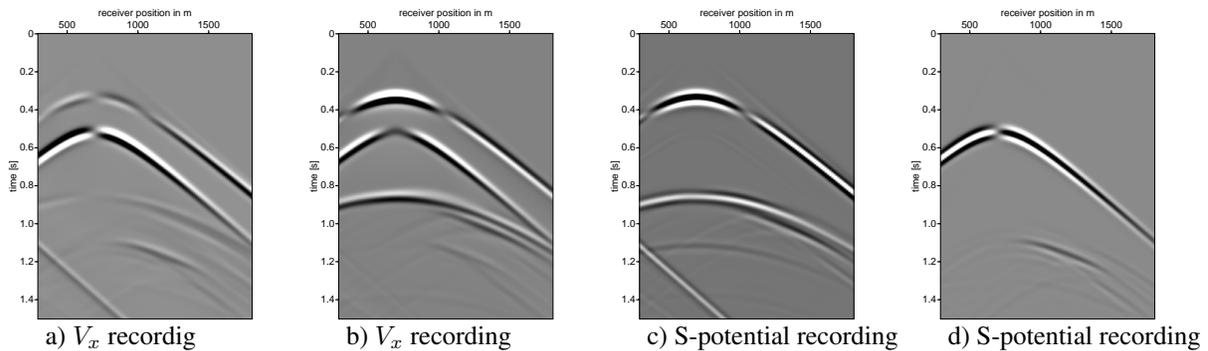
Figure 19: *Panel a,b, c and d show different types of receiver measurements in a visco-elastic medium. Note that the direct field has been subtracted from the recordings.*

7.3 Example for different source distributions: `fdelmodc_sourcepos.scr`

Figure 20 shows recorded data for different kind of source distributions and different source signatures. The used source distributions are random source positions between $500 \leq z \leq 4100$, random positions between $2700 \leq z \leq 4100$, and a plane layer at $z = 2700$ m. For the source positions 800 sources are used, for the sources positioned on the vertical plane (20c and f) on every grid point a source is placed. Two types of source signatures are used; a Ricker wavelet with a frequency peak at 10 Hz, and random source signals with a maximum frequency of 30 Hz.

The script file `demo/fdelmodc_sourcepos.scr` show how these 6 modeled shots can easily be modeled. The script illustrates how to define different source distributions within the program and how a random source signature is defined. These kind of experiments can be used to investigate the sensitivity of seismic interferometry on different source distributions (Thorbecke and Draganov, 2011). Each modeling takes about 100 seconds making the total runtime of the script about 10 minutes.

7.4 Example with receivers on a circle: `fdelmodc_circ.scr`

Receivers can be placed on a circle using the parameters `rrcv=` for the radiation of the circle, `oxrcv,ozrcv` sets the origin (in meter, not grid points) of the circle and `dphi` the distance between the receivers in angle. Note that due to the grid of the wavefields the receivers are not placed on exactly a circle, but to the closest grid point. The option `rec_int_vx` or `rec_int_vz` set to 3 will interpolate (bi-linear) the fields to the exact position of the receivers on the circle. When `rrcv` is used the parameters `rec_int_vx` and `rec_int_vz` are automatically set to 3. The result of using this option, in a homogeneous background medium with a contrast placed in the middle of the model, is shown in Figure 21. The runtime of this script is 40 seconds.

a) random $500 \leq z \leq 4100$ b) random $z \leq 2700$ c) plane $z = 2700$

d) random $500 \leq z \leq 4100$ e) random $z \leq 2700$ f) plane $z = 2700$
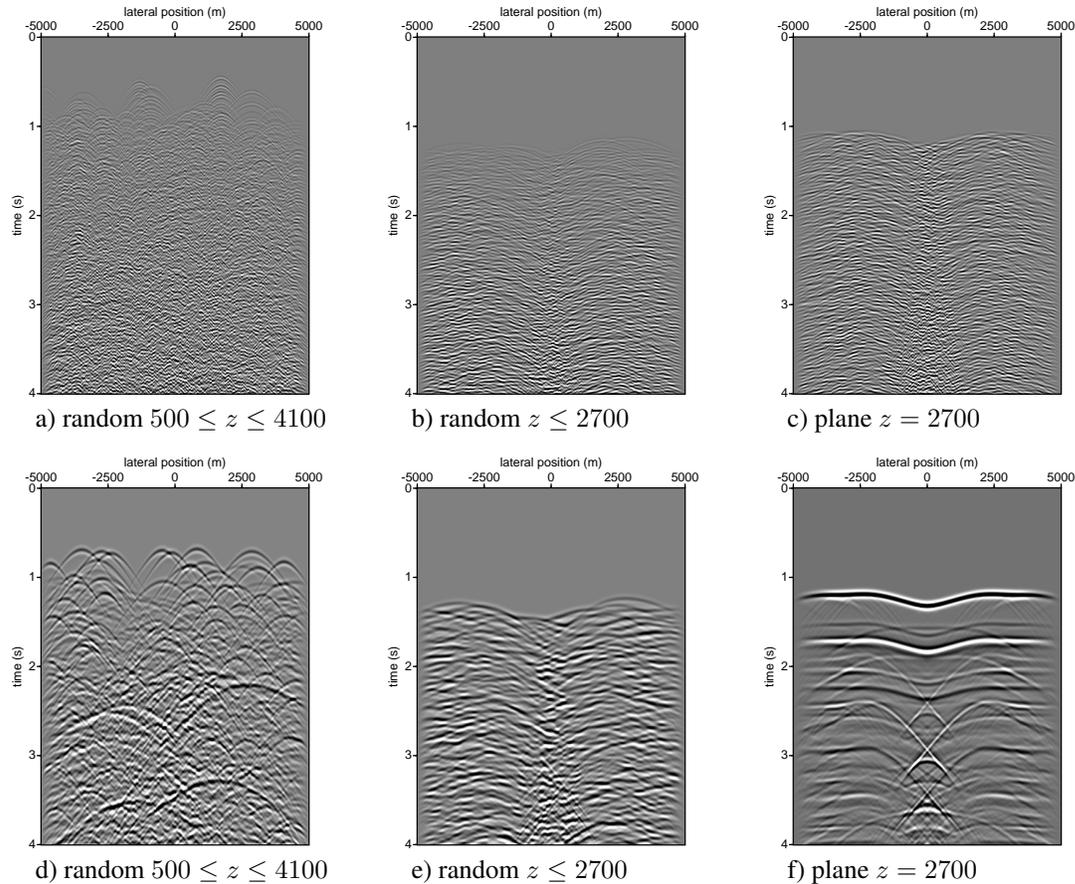
Figure 20: *To investigate the type of noise present in the reconstructed reflection from seismic interferometry different experiments have been carried out. The first 4 seconds of different types of source signatures and source distribution are shown in a to f. Noise signatures are used in a,b and c and a Ricker wavelet is used on d,e and f. The sources are distributed in the area indicated in the caption of the pictures: for x-positions in the indicated z-range.*

7.5 Example with topography: `fdelmodc_topgraphy.scr`

To make a more complicated a topography can be included. The receivers are placed on this topography. A directly modeled result is generated with an active source in the middle of the model and is shown in Figure 22. To include topography in a model the c_p velocity must be chosen zero in the layer above the topography. Note that the density must not be set to zero. The source and receiver positions are then given on a horizontal surface above the defined topography. The code then searches for the first non-zero c_p value below the horizontal surface. In this way the source and receiver positions are placed at the first non-zero c_p position. The parameter `sinkdepth=` can be used to place the source and receiver `sinkdepth` grid positions deeper than the first non-zero velocity. The command how to make the model and how the shot record is computed is shown below and can be found in the script file `demo/fdelmodc_topography.scr`. The runtime of this script is 500 seconds.

Note that the receiver (not the source) positions can also be sinked through a layer, with a non-zero velocity, to the next interface when the parameter `sinkvel=` is defined . The receivers must then be placed somewhere in that layer and only works for homogeneous layers. An example of usage is to model OBC data where the receivers have to be placed on the topography of the sea-bottom and the sources are close to the free surface. Using `sinkvel=1` will place the receivers at the bottom of the sea layer. The script `fdelmodc_topgraphy.scr` can also be used to generate OBC data. In the script you then have to change in `makemod cp0=1500` and in fdelmodc add the parameter `sinkvel=1`.

```
makemod sizex=10000 sizez=4100 dx=5 dz=5 cp0=0 ro0=1000 file_base=real2.su \
    orig=0,-800 gradunit=0 \
    intt=def poly=2 cp=2450 ro=1000 gradcp=14 grad=0 \
    x=0,1000,1700,1800,2000,3000,4000,4500,6000,6800,7000,7500,8100,8800,10000 \
```
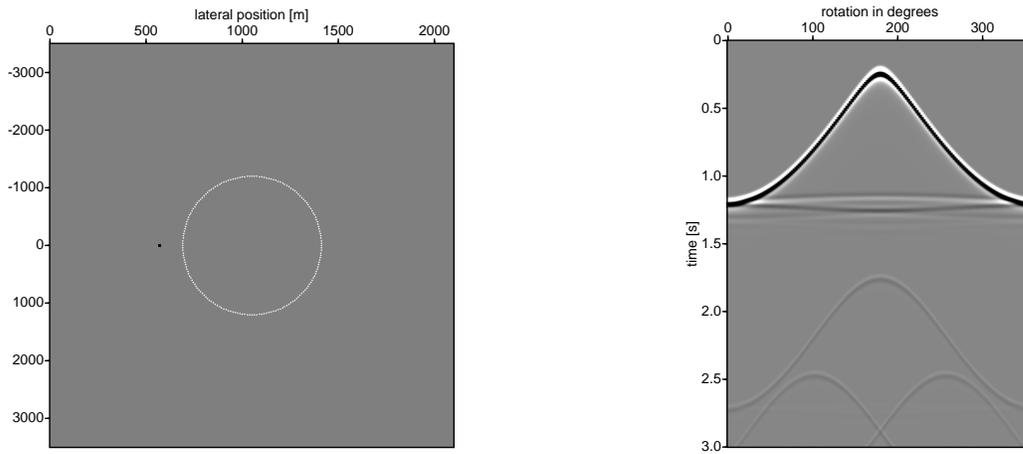
Figure 21: The middle of the model contains a circular contrast and the receiver are placed on a circle around this contrast. The left picture show the source (black dot) and receiver (white dots) positions. Right is the recorded wavefield at the receiver positions.

```
    z=-100,-200,-250,-200,-200,-120,-300,-600,-650,-500,-350,-200,-200,-150,-200  \
    intt=rough var=200,3.2,1 poly=2 x=0,3000,8000,10000 \
            z=400,250,300,500 cp=4500,4200,4800,4500 ro=1400 gradcp=5 grad=0 \
    intt=def poly=2 x=0,2000,3000,5000,7000,8000,10000 \
        z=1100,1100,1100,1600,1100,1100,1100 cp=4000 ro=2000 gradcp=8 grad=0 \
    intt=def poly=0 x=0,10000 z=1750,2050 cp=4500,5100 ro=1500 gradcp=13 grad=0 \
    intt=def poly=0 x=0,10000 z=1850,2150 cp=6000,4200 ro=1500 gradcp=14 grad=0 \
    intt=def poly=0 x=0,10000 z=1950,2250 cp=4800,4800 ro=1500 gradcp=5 grad=0 \
    intt=def poly=0 x=0,10000 z=2000,2300 cp=6100,5000 ro=1500 gradcp=13 grad=0 \
    intt=def poly=0 x=0,10000 z=2100,2400 cp=3800,5000 ro=1500 gradcp=20 grad=0 \
    intt=def poly=0 x=0,10000 z=2150,2450 cp=5000 ro=1500 gradcp=14 grad=0 \
    intt=def poly=0 x=0,10000 z=2350,2650 cp=5800 ro=1500 gradcp=5 grad=0 \
    intt=def poly=0 x=0,10000 z=2600,2600 cp=5500 ro=2200 gradcp=5 grad=0

fdelmodc \
    file_cp=vel2_edge_cp.su ischeme=1 \
    file_den=vel2_edge_ro.su \
    file_rcv=shot_real2_x5000_topo.su \
    file_src=G2.su \
    dtrcv=0.004 \
    verbose=4 \
    tmod=3.004 \
    dxrcv=20.0 \
    zrcv1=-800 \
    xrcv1=0 \
    xrcv2=10000 \
    sinkdepth=1 \
    src_random=0 \
    wav_random=0 \
    xsrc1=5000 \
    xsrc2=5000 \
    zsrc1=-800 \
    tsrc1=0.0 \
    dipsrc=1 \
    ntaper=$ntap \
    left=4 right=4 bottom=4 top=1
```
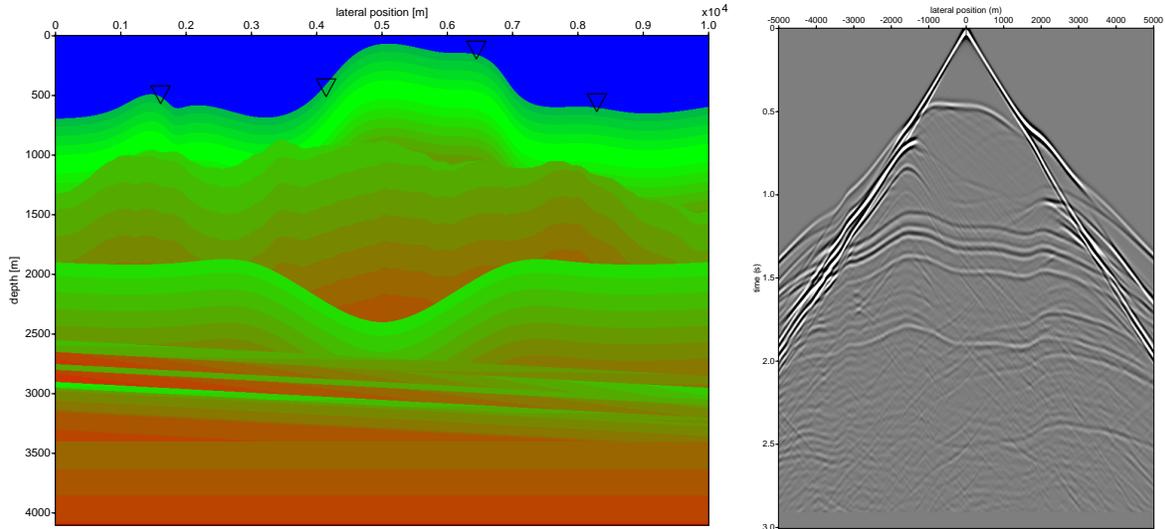
Figure 22: The receivers are placed from -5000 m to 5000 m at every 20 m on the topography. The modeled reflection response, for a source at 5000 m placed on the the topography, is shown in the right picture.

7.6 Example verification with analytical results: `FigureGreenDxAppendixA.scr`

7.6.1 Acoustic

To verify the accuracy and the correctness of the FD program we have compared the finite-difference calculation of a Green's function in a homogenous acoustic medium with the analytical Green's function. Four analytical Green's functions have been used for verification using:

- monopole source and pressure (P) receivers,

- monopole source and vertical particle-velocity (V_z) receivers,

- dipole source and P receivers,

- dipole source and V_z receivers.

The wave equation solved by the FD program implements the following source term for volume injection sources (parameter `src_injectionrate=0`, which is the default setting):

$$\rho\delta(\mathbf{x} - \mathbf{x}_s)\delta(t) \tag{118}$$
$$\rho\delta(\mathbf{x} - \mathbf{x}_s)$$

and for volume injection rate sources (parameter `src_injectionrate=1`):

$$\rho\delta(\mathbf{x} - \mathbf{x}_s)\frac{\partial\delta(t)}{\partial t} \tag{119}$$
$$j\omega\rho\delta(\mathbf{x} - \mathbf{x}_s)$$

The analytical Green's functions for volume injection sources are given by (Berkhout, 1987, page 141-146):

$$\hat{G}^{p,p}(\mathbf{x}, \mathbf{x}_s) = P^{mon} = \frac{-j\rho}{4}H_0^{(2)}(kr), \tag{120}$$

$$\hat{G}^{v,p}(\mathbf{x}, \mathbf{x}_s) = V_z^{mon} = \frac{cos(\phi)}{4c}H_1^{(2)}(kr), \tag{121}$$

$$\hat{G}^{p,v}(\mathbf{x}, \mathbf{x}_s) = P^{dip} = j\omega\frac{cos(\phi)}{4c}H_1^{(2)}(kr), \tag{122}$$

$$\hat{G}^{v,v}(\mathbf{x}, \mathbf{x}_s) = V_z^{dip} = \frac{-kcos^2(\phi)}{4\rho c}H_0^{(2)}(kr) - \frac{k(1 - 2cos^2(\phi))}{4\rho ckr}H_1^{(2)}(kr), \tag{123}$$

44
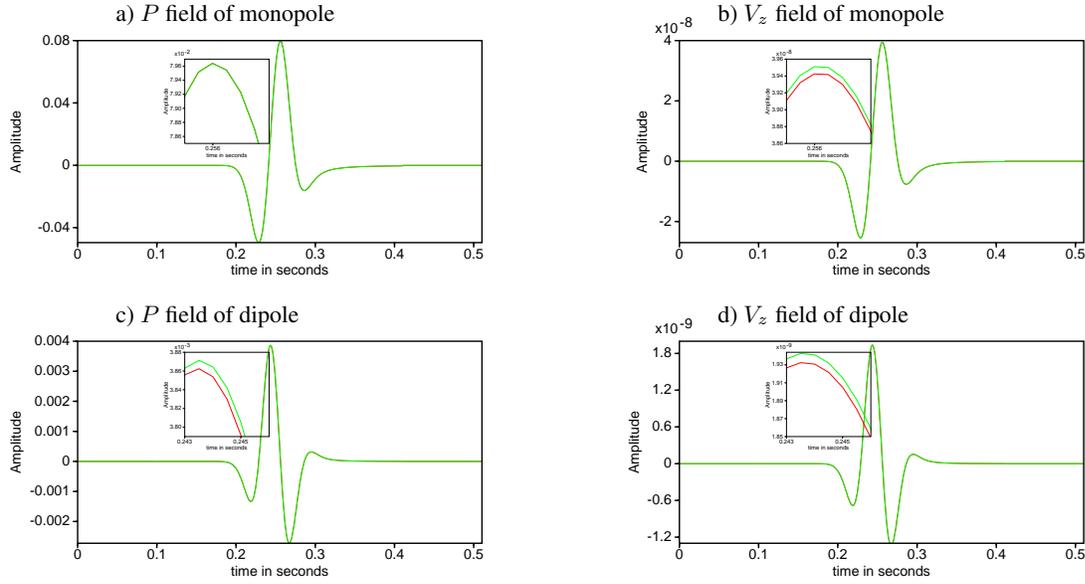
Figure 23: Comparison of Green's functions in an acoustic homogeneous medium for monopole (top) and dipole sources (bottom) with pressure (P) and particle velocity (V_z), left and right, respectively, recorded wavefields. The onsets show the differences for the positive peak of the wavelet, the lower line represents the finite-difference result. The script `FigureGreenAppendixA.scr` in the FiguresPaper directory calculates the data and reproduces the pictures.

where

$$H_0^{(2)}(kr) = J_0(kr) - jY_0(kr), \tag{124}$$

$$H_1^{(2)}(kr) = J_1(kr) - jY_1(kr), \tag{125}$$

$$r = \sqrt{(x^2 + (z_s - z_r)^2)}, \tag{126}$$

$$cos(\phi) = \frac{|z_s - z_r|}{r}, \tag{127}$$

x represents the lateral distance and z_s and z_r are the depth positions of the source and receiver, respectively. J_0 and J_1 are the Bessel functions of the first kind of orders 0 and 1, respectively. Y_0 and Y_1 are the Bessel functions of the second kind of orders 0 and 1, respectively. The wavenumber $k = \omega/c$, where c is the velocity of the medium. The analytical responses are generated by the program 'green', included in the source code distribution in the `utils` directory. In the beginning of section 6.5 it is explained that an injection source is implemented (and not injection rate). Sometimes in literature equations 120-123 have an extra factor $j\omega$ when injection rates are used.

Note that in 2D media the far field expression of the Hankel functions in equations (120)-(123) contain a 45 degree phase-shift. This phase shift is of course also present in the computed FD results. For example if you have a Ricker wavelet as input source (e.g. modelling in a homogeneous medium), then the recorded wave will have a 45 degree phase shift compared to the input Ricker wavelet. Another way to see this is, is to realise that a point source in 2D is represented by a line source in 3D. For a more detailed explanation see also (Berkhout, 1987, page 141-146).

In the staggered-grid implementation, the P- and V_z-fields are positioned at different spatial grids and the V_z fields have been interpolated to the P-field grid position to be able to compare them with the analytical solution positioned at the P-field position. The FD scheme is also staggered in time and the modeled P-field is shifted half a time step compared to the V_z-field. For the implementation of a dipole source, two grid positions are used and this gives an extra time delay of $\frac{0.5\Delta z}{c}$, where c is the velocity at the source position and Δz the discretization step in the z-direction. In the comparison with the analytical Green's functions these discretization effects have all been taken into account.

The reference medium has a velocity of 2000 m/s and a density of 1000 kg/m^3. The source is positioned 500 m below the receiver. For the finite-difference code a spatial grid of 2.5 m and a time step of 0.5 ms has been used to
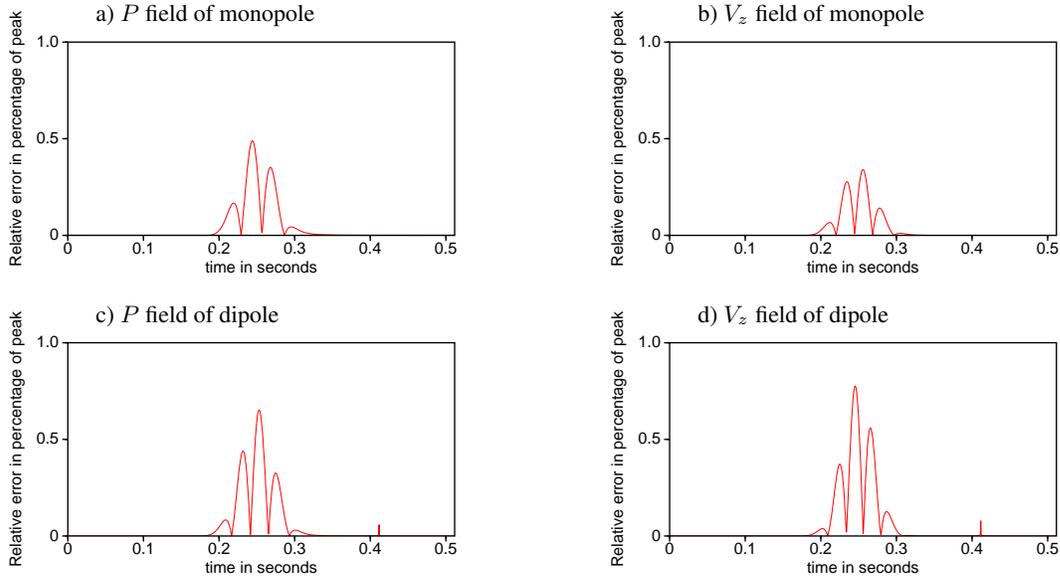
Figure 24: Difference between the analytical Green's function and the finite-difference result in an acoustic medium for monopole and dipole sources for P and V_z recorded fields. The difference is shown as percentage of the maximum peak in the analytical Green's function. A grid spacing of $\Delta = 2.5$ m is used. The script FigureGreenDxAppendixA.scr in the FiguresPaper directory calculates the data and reproduces the pictures.

compute the results.

The comparison between the analytical Green's functions and the finite-difference computed results are shown in Figure 23 for the four different configurations mentioned. The curves are perfectly overlapping and only after zooming in at the peak of the wavelet it is possible to observe differences. The difference between the analytical Green's function and the FD results is less then 1% of the peak of the analytical Green's function.

The script to reproduce the verification figures can be found in FiguresPaper directory (FigureGreenAppendixA.scr). The difference between the analytical Greens function and the finite-difference result is less then 1% and shown in Figure 24 for the same Green's functions shown in Figure 23. The small peak at 0.4115 seconds is caused by the program to compute the shift of -0.1 seconds and caused by the the limited number of time samples. This shift puts the peak of the input wavelet at $t = 0$. To have the wavefields on the same grid the V_z grid has been interpolated to the P grid and also an extra time shift of a half Δt has been used to compensate for the staggering in time.

The difference between the analytical Green's function and the finite-difference result using a smaller grid spacing will give smaller errors and larger grid spacing will give larger errors. This is shown in Figure 25. Note that for $\Delta = 5$ dispersion is staring to become visible in the difference plot, but the error is still smaller than 1.2%. The script to reproduce the pictures of Figure 25 can be found in the fdelmodc/FiguresPaper directory (FigureGreenDxAppendixA.scr). The FD program takes 60 seconds to model the result using a grid spacing of $\Delta = 5$, 200 seconds with $\Delta = 2.5$ and 800 seconds with $\Delta = 1.0$.
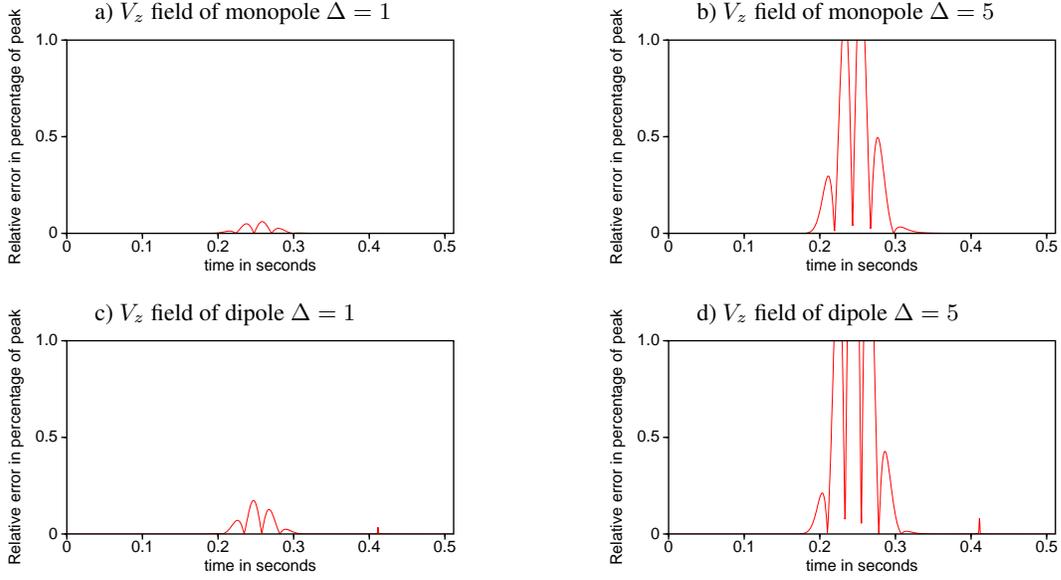
Figure 25: Difference between the analytical Green's function and the finite-difference result in an acoustic medium for monopole and dipole sources for V_z recorded fields using a grid spacing of 1 and 5 meter. The difference is shown as percentage of the maximum peak in the analytical Green's function. The larger errors for the $\Delta = 5$ model results indicates that dispersion starts to develop. The script `FigureGreenDxAppendixA.scr` in the FiguresPaper directory calculates the data and reproduces the pictures.

7.6.2 Elastic

This section is made together with Karel van Dalen, who has derived the Green's function in a homogenous elastic medium.

To verify the accuracy and the correctness of the FD program we have compared the finite-difference calculation of a Green's function in a homogenous elastic medium with the analytical Green's function. Four analytical Green's functions have been used for verification :

- force source in x-direction source and vertical particle-velocity (V_z) receivers (`Fx=1`),

- force source in z-direction source and vertical particle-velocity (V_z) receivers (`Fz=1`),,

- deformation source in x-direction source and vertical particle-velocity (V_z) receivers (`Tx=1`),

- deformation source in z-direction source and vertical particle-velocity (V_z) receivers (`Tz=1`).

General Green's functions:

$$\hat{G}_{i,j}^{v,f}(\mathbf{x},\mathbf{x}_s) = \frac{j\omega}{\rho}\{\frac{1}{c_s^2}\delta_{ij}\hat{G}_s(\mathbf{x},\mathbf{x}_s) - \frac{1}{\omega^2}\partial_i\partial_j(\hat{G}_p(\mathbf{x},\mathbf{x}_s) - \hat{G}_s(\mathbf{x},\mathbf{x}_s))\} \tag{128}$$

$$\hat{G}_p(\mathbf{x},\mathbf{x}_s) = \frac{-j}{4}H_0^{(2)}(k_p r), \text{with} \quad k_p = \frac{\omega}{c_p} \tag{129}$$

$$\hat{G}_s(\mathbf{x},\mathbf{x}_s) = \frac{-j}{4}H_0^{(2)}(k_s r), \text{with} \quad k_s = \frac{\omega}{c_s} \tag{130}$$

$$\partial_i\partial_j\hat{G}(\mathbf{x},\mathbf{x}_s) = \frac{-j}{4}\{-\frac{\bar{x}_i\bar{x}_j}{r^2}k^2H_0^{(2)}(kr) + k(2\frac{\bar{x}_i\bar{x}_j}{r^3} - \frac{\delta_{ij}}{r})H_1^{(2)}(kr)\} \tag{131}$$

$$\partial_z\partial_z\hat{G}(\mathbf{x},\mathbf{x}_s) = \frac{-j}{4}\{-\frac{\bar{z}^2}{r^2}k^2H_0^{(2)}(kr) + k(2\frac{\bar{z}^2}{r^3} - \frac{1}{r})H_1^{(2)}(kr)\} \tag{132}$$

$$\partial_z\partial_x\hat{G}(\mathbf{x},\mathbf{x}_s) = \frac{j\cos(\phi)\bar{x}}{4r}\{k^2H_0^{(2)}(kr) - \frac{2k}{r}H_1^{(2)}(kr)\} \tag{133}$$

$$\tag{134}$$

The corresponding analytical Green's functions are given by:

$$\hat{G}^{v,f_x}(\mathbf{x}, \mathbf{x}_s) = -\frac{j}{\omega \rho}\{\partial_z \partial_x(\hat{G}_p(\mathbf{x}, \mathbf{x}_s) - \hat{G}_s(\mathbf{x}, \mathbf{x}_s))\}, \tag{135}$$

$$= \frac{\bar{z}\bar{x}}{4\omega \rho r^2}\{(k_p^2 H_0^{(2)}(k_p r) - \frac{2k_p}{r}H_1^{(2)}(k_p r)) - (k_s^2 H_0^{(2)}(k_s r) - \frac{2k_s}{r}H_1^{(2)}(k_s r))\} \tag{136}$$

$$\hat{G}^{v,f_z}(\mathbf{x}, \mathbf{x}_s) = \frac{j\omega}{c_s^2 \rho}\hat{G}_s(\mathbf{x}, \mathbf{x}_s) - \frac{j}{\omega \rho}\{\partial_z \partial_z(\hat{G}_p(\mathbf{x}, \mathbf{x}_s) - \hat{G}_s(\mathbf{x}, \mathbf{x}_s))\}, \tag{137}$$

$$= \frac{\omega}{4c_s^2 \rho}H_0^{(2)}(k_s r) \tag{138}$$

$$- \frac{k_p}{4\omega \rho r^2}\{-\bar{z}^2 k_p H_0^{(2)}(k_p r) + (\frac{\bar{z}^2 - \bar{x}^2}{r})H_1^{(2)}(k_p r)\} \tag{139}$$

$$+ \frac{k_s}{4\omega \rho r^2}\{-\bar{z}^2 k_s H_0^{(2)}(k_s r) + (\frac{\bar{z}^2 - \bar{x}^2}{r})H_1^{(2)}(k_s r)\} \tag{140}$$

$$\tag{141}$$

$$\hat{G}^{v,\tau_{xz}}(\mathbf{x}, \mathbf{x}_s) = \tag{142}$$

$$\tag{143}$$

$$\hat{G}^{v,\tau_{zz}}(\mathbf{x}, \mathbf{x}_s) = \tag{144}$$

$$\tag{145}$$

where

$$H_0^{(2)}(kr) = J_0(kr) - jY_0(kr), \tag{146}$$

$$H_1^{(2)}(kr) = J_1(kr) - jY_1(kr), \tag{147}$$

$$\bar{x} = x - x_s, \tag{148}$$

$$\bar{z} = z - z_s, \tag{149}$$

$$r = \sqrt{(x - x_s)^2 + (z - z_s)^2}, \tag{150}$$

$$r = \sqrt{(\bar{x}^2 + \bar{z}^2)}, \tag{151}$$

$$cos(\phi) = \frac{|z - z_s|}{r}, \tag{152}$$

x represents the lateral distance and z_s and z are the depth positions of the source and receiver, respectively. J_0 and J_1 are the Bessel functions of the first kind of orders 0 and 1, respectively. Y_0 and Y_1 are the Bessel functions of the second kind of orders 0 and 1, respectively. The wavenumber $k = \omega/c$, where c is the velocity of the medium. The analytical responses are generated by the program 'green', included in the source code distribution in the `utils` directory.

A   Source and directory structure

```
  /
  Make_include  ............File with system specific setting and can be adapted for specific Unix systems
  Makefile  ...................Controls the compilation and linking of the programs in the subdirectories
  README
  SU_LEGAL_STATEMENT.txt
  bin  .....................................Directory for the binaries compiled and linked using the Makefile
  FFTlib  .................................Library for FFT transformation routines used by the programs
  fdelmodc  ...........................This directory contains all source code for the program fdelmodc
```

B   Differences in parameter use compared with DELPHI's fdacmod

fdelmodc uses many similar parameters as the DELPHI application fdacmod. The differences are:

- acoustic modeling → `ischeme=1`

- `file_vel=` → `file_cp=`

- `file_att=` → `file_qp=` or `Qp=`

- `dipsrc=0` → `src_orient=1`

- `dipsrc=1` → `src_orient=2`

- `diprcv=0` → `rcv_type_p=1`

- `diprcv=1` → `rcv_type_vz=1`

- `xrcv=` → `xrcva=`

- `zrcv=` → `zrcva=`

In fdelmodc there is no `dxsrc dzsrc dxspread dzspread tapfact`.

C Makewave

The wavelets generated with makewave are

- g0: Gaussian wavelet

$$G_0(f) = \exp(-\frac{f^2}{f_p^2}) \tag{153}$$

- g1: first derivative of a Gaussian wavelet

$$G_1(f) = \frac{f}{\sqrt{2}f_p} \exp(-\frac{f^2}{2f_p^2}) \tag{154}$$

- g2: second derivative of a Gaussian wavelet

$$G_2(f) = \frac{f^2}{f_p^2} \exp(-\frac{f^2}{f_p^2}) \tag{155}$$

where f_p is the peak in the spectrum of the wavelet.

References

Alford, R., Kelly, K., and Boore, D. (1974). Accuracy of nite-difference modeling of the acoustic wave equation. *Geophysics*, 39(6):834–842.

Bauer, A. L., Loubère, R., and Wendroff, B. (2008). On stability of staggered schemes. *SIAM J. Numer. Anal.*, 46(2):996–1011.

Berkhout, A. J. (1987). *Applied seismic wave theory*. Elsevier, Amsterdam.

Bohlen, T. (2002). Parallel 3-d viscoelastic nite difference seismic modelling. *Computer and Geosciences*, 28:887–899.

Chew, W. C. and Liu, Q. H. (1996). Perfectly matched layers for elastodynamics: A new absorbing boundary condition. *Journal of Computational Acoustics*, pages 341–359.

Courant, R., Friedrichs, K., and Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM Journal, English translation of the 1928 German original*, pages 215–234. Available as download `http://www.stanford.edu/class/cme324/classics/courant-friedrichs-lewy.pdf`.

Drossaert, F. H. and Giannopoulos, A. (2007). A nonsplit complex frequency-shifted pml based on recursive integration for fdtd modeling of elastic waves. *Geophysics*, 72(2):T9–T17.

Fornberg, B. (1988). Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51(184):699–706.

Pérez-Ruiz, J. A., Luzón, F., and García-Jerez, A. (2005). Simulation of an irregular free surface with a displacement finite-difference scheme. *Bulletin of the Seismological Society of America*, 95(6):2216–2231.

Robertsson, J. O. A. (1996). A numerical free-surface condition for elastic/viscoelastic finite-difference modeling in the presence of topography. *Geophysics*, 61:1921–1934.

Robertsson, J. O. A., Blanch, J. O., and Symes, W. W. (1994). Viscoelastic finite-difference modeling. *Geophysics*, 59(09):1444–1456.

Saenger, E. H. and Bohlen, T. (2004). Finite-difference modeling of viscoelastic and anisotropic wave propagation using the rotated staggered grid. *Geophysics*, 69(2):583–591.

Sei, A. (1995). A family of numerical schemes for the computation of elastic waves. *SIAM J. Sci. Comput.*, 16(4):898–916.

Sei, A. and Symes, W. (1995). Dispersion analysis of numerical wave propagation and its computational consequences. *Journal of Scientific Computing*, 10(1):1–27.

Thorbecke, J. and Draganov, D. (2011). Finite-difference modeling experiments for seismic interferometry. *Geophysics*, 76(6):H1–H18.

van Vossen, R., Robertsson, J. O. A., and Chapman, C. (2002). Finite-difference modeling of wave propagation in a fluid-solid configuration. *Geophysics*, 67(2):618–624.

Virieux, J. (1986). P-Sv wave propagation in heterogeneous media - Velocity-stress finite-difference method. *Geophysics*, 51(04):889–901.

Wapenaar, K. (1998). Reciprocity properties of one-way propagators. *Geophysics*, 63(4):17951798.